การติดตั้งและปรับแต่งลินุกซ์

เทพพิทักษ์ การุญบุญญานันท์

สารบัญ

บทที่

1

พาร์ทิชันและการบูต

1.1 โครงสร้างข้อมูลของฮาร์ดดิสก์

การใช้เนื้อที่ในฮาร์ดดิสก์จะไม่ได้เป็นระบบไฟล์เดียวทั้งแผ่นเหมือนฟลอปปี้ เพราะฮาร์ดดิสก์มีขนาดใหญ่กว่ามาก จึงสามารถแบ่งเป็นหลายระบบไฟล์แยกกัน การเตรียมเนื้อที่สำหรับเก็บข้อมูลจึงต้องเริ่มจากการแบ่งพาร์ทิชันรองรับระบบไฟล์ต่างๆ ก่อนเสมอ ถึงแม้จะใช้เพียงระบบไฟล์เดียวก็ตาม (ซึ่งไม่คุ้มแล้วสำหรับขนาดฮาร์ดดิสก์ในปัจจุบัน)

เมื่อคุณเปิดเครื่อง CPU ของคุณยังไม่มีโปรแกรมอะไรให้ทำ ข้อกำหนดของ CPU ก็คือ จะเริ่ม execute โปรแกรมที่แอดเดรส FFFF0h เสมอ และก็เป็นแอดเดรสเริ่มต้นของ BIOS ที่จะตรวจสอบระบบ ตั้งค่าเริ่มต้น ตรวจหาอุปกรณ์ต่างๆ แล้วพยายามบูตระบบปฏิบัติการตามลำดับที่ตั้งไว้ใน CMOS

สำหรับฮาร์ดดิสก์นั้น ข้อตกลงระหว่าง BIOS ต่างๆ ของเครื่อง PC เรื่องโครงสร้างในฮาร์ดดิสก์ก็คือ เซกเตอร์แรกของฮาร์ดดิสก์ (cylinder 0, head 0, sector 1) ซึ่งเรียกว่า *Master Boot Record (MBR)* จะเก็บข้อมูลต่อไปนี้

- *Master Partition Table* เป็นข้อมูลของพาร์ทิชันต่างๆ ในฮาร์ดดิสก์ ซึ่งมีเนื้อที่ให้เก็บเพียง 4 พาร์ทิชันเท่านั้น
- *Master Boot Code* เป็นโปรแกรมเล็กๆ ที่ BIOS จะโหลดเพื่อบูตระบบต่อไป โดยปกติโปรแกรมนี้จะเลือกพาร์ทิชันที่จะบูตแล้วส่งต่อให้โปรแกรมบูตในพาร์ทิชันนั้นๆ ดำเนินการโหลดระบบปฏิบัติการต่อไป

1.2 พาร์ทิชัน

ด้วยโครงสร้างของ Master Partition Table ทำให้ฮาร์ดดิสก์ของเครื่อง PC สามารถมีพาร์ทิชันจริงซึ่งเรียกว่า *primary partition* ได้ไม่เกิน 4 พาร์ทิชัน แต่ก็สามารถขยายเพิ่มได้ โดยกำหนดให้พาร์ทิชันหนึ่งเป็น *extended partition* ซึ่งจะสามารถบรรจุ *logical partition* ได้อีกไม่จำกัดจำนวน เนื่องจากใช้โครงสร้างแบบ linked list ในการเชื่อมระหว่าง logical partition ต่างๆ นั่นเอง (แต่ถ้าคุณใช้ดอส

หรือวินโดวส์ ก็จะจำกัดจำนวน primary partition และ logical partition รวมกันในฮาร์ดดิสก์ทุกลูกแล้วได้มากที่สุดไม่เกิน 24 พาร์ทิชัน เนื่องจากการใช้ตัวอักษรภาษาอังกฤษแทนไดรฟ์ต่างๆ โดยเริ่มเรียกฮาร์ดดิสก์จาก `C:` นั่นเอง)

ในมุมมองของซอฟต์แวร์แล้ว จะไม่มีความแตกต่างการใช้งานระหว่าง primary partition และ logical partition เลย ยกเว้นว่า MBR ของดอสจะสามารถบูตจาก primary partition ได้เท่านั้น ไม่สามารถบูตจาก logical partition ได้ ดังนั้น โปรแกรม `fdisk` ของดอส/วินโดวส์จึงไม่ยอมให้เซ็ตให้ logical partition เป็น bootable (active) partition แต่ `fdisk` ของ GNU/Linux ไม่มีข้อจำกัดนี้ แต่ก็หมายความว่า คุณต้องใช้ boot loader ที่เหมาะสมด้วย (เช่น LILO หรือ GRUB)

ลินุกซ์จะกำหนด device ให้กับฮาร์ดดิสก์ต่างๆ ที่ต่อกับเครื่องด้วยรูปแบบ

```
/dev/[hs]d[a-z]
```

โดยที่อักษรตัวแรก (`h` หรือ `s`) จะแยกอุปกรณ์ IDE (`h`) และ SCSI (`s`) ออกจากกัน ส่วนตัวอักษรตัวท้าย (`a-z`) จะแทนอุปกรณ์ที่ 1, 2, . . . ตามลำดับ

สำหรับอุปกรณ์ IDE นั้น mainboard ปัจจุบันสามารถต่ออุปกรณ์ได้สูงสุด 4 ตัว และลินุกซ์จะมองผ่าน device ต่างๆ ดังนี้

`/dev/hda` แทน primary master

`/dev/hdb` แทน primary slave

`/dev/hdc` แทน secondary master

`/dev/hdd` แทน secondary slave

ซึ่งอุปกรณ์เหล่านี้อาจเป็น IDE CD-ROM drive ก็ได้

ส่วนอุปกรณ์ SCSI นั้น ต่อพ่วงต่อๆ กันได้ไม่จำกัด

จากนั้น พาร์ทิชันต่างๆ ในฮาร์ดดิสก์แต่ละลูกจะถูกกำหนด device โดยเติมตัวเลขต่อท้าย device ของฮาร์ดดิสก์ เลข 1–4 จะแทน primary/extended partition และเลข 5 ขึ้นไปจะแทน logical partition ถึงแม้จะมี primary/extended partition ไม่ครบ 4 ก็ตาม ตัวอย่างเช่น

`/dev/hda1` แทน primary partition แรกของ primary master IDE hard disk

`/dev/hdb5` แทน logical partition แรกของ primary slave IDE hard disk

`/dev/hda9` แทน logical partition ที่ 5 ของ primary master IDE hard disk

1.3 การบูต

เมื่อ BIOS ค้นหาอุปกรณ์สำหรับบูตตามลำดับจนพบอุปกรณ์แรกที่บูตได้ BIOS ก็จะโหลดเซกเตอร์แรกขึ้นมา execute สำหรับฮาร์ดดิสก์แล้ว ก็คือ MBR นั่นเอง โดย BIOS จะ execute Master Boot Code ซึ่งจะดำเนินการโหลดระบบปฏิบัติการต่อไป

สำหรับ MBR ของดอสปกติแล้ว หลังจากที่อ่านตารางพาร์ทิชันและตรวจสอบ extended partition ทั้งหมดแล้ว ก็จะโอนการบูตต่อให้กับ boot sector ของพาร์ทิชันที่ active ซึ่งจะมีได้เพียงหนึ่งพาร์ทิชันเท่านั้น ทำให้บูตได้เพียงระบบเดียว แต่ Windows NT และ OS/2 จะเริ่มมีโปรแกรม boot manager ที่สามารถเลือกพาร์ทิชันที่จะบูตได้มาให้ด้วย เพื่อรองรับการติดตั้งระบบเหล่านั้นร่วมกับ Windows 9x ชนิดอื่นในเครื่องเดียวกันนั่นเอง

สำหรับลินุกซ์แล้ว ก็มี boot loader ที่เป็นซอฟต์แวร์เสรีให้เลือกใช้อยู่จำนวนหนึ่ง ที่ใช้กันแพร่หลายก็ได้แก่ LILO (LInux LOader) และ GRUB (GRand Unified Boot loader) ซึ่งสามารถติดตั้งได้

หลายรูปแบบ และสามารถบูตระบบปฏิบัติการได้หลากหลาย ไม่ว่าจะเป็น MS-DOS, Windows ME, Windows 2000, Windows XP, GNU/Linux, FreeBSD, NetBSD, OpenBSD, GNU/Hurd

boot loader จะแทรกตัวเข้าไปในขั้นตอนการบูต อาจจะด้วยการเปลี่ยน MBR โดยตรง หรืออาจจะอยู่ใน boot sector ของพาร์ทิชันต่างหากที่ถูกเซ็ตให้ active ก็ได้ boot loader จะมีข้อมูลของระบบปฏิบัติการในพาร์ทิชันต่างๆ เก็บไว้ (บางโปรแกรมอาจมีฟังก์ชันสำหรับตรวจสอบพาร์ทิชันก็ได้) และจะรับ input จากผู้ใช้ว่าจะบูตพาร์ทิชันไหน ก่อนที่จะโหลดระบบปฏิบัติการจากพาร์ทิชันนั้นๆ ขึ้นมาทำงาน

ทั้งนี้ ในการโหลดระบบปฏิบัติการ ก็จำเป็นต้องอ่านไฟล์ kernel image ขึ้นมาจากระบบไฟล์ของแต่ละระบบได้ หรือสามารถส่งงานต่อให้ loader ของระบบนั้นๆ ได้ ซึ่งเป็นเรื่องที่ค่อนข้างอาศัยเทคนิคการจัดการพอสมควร

จากฟังก์ชันต่างๆ ที่กล่าวมา ทำให้ boot loader มักมีขนาดใหญ่เกินกว่าจะบรรจุใน MBR หรือ boot sector ของพาร์ทิชันได้ จึงจำเป็นต้องแบ่ง loader ออกเป็นสองส่วน ส่วนที่อยู่ใน MBR หรือ boot sector จะเป็นโค้ดเล็กๆ ที่โหลดส่วนที่เหลือจากพาร์ทิชันต่างหาก

1.4 การแบ่งพาร์ทิชันสำหรับลินุกซ์

การติดตั้งลินุกซ์ทั้งหมดในพาร์ทิชันเดียวเป็นสิ่งที่เป็นไปได้ แต่ไม่ใช่สิ่งที่น่าทำ ปัจจัยต่างๆ บางส่วนที่จะมีผลต่อการตัดสินใจในการแบ่งพาร์ทิชัน ได้แก่

1.4.1 ขนาดของ Virtual Memory

ลินุกซ์ระบบ virtual memory เหมือนยูนิกซ์ทั่วไปในการขยายเนื้อที่หน่วยความจำเพิ่มจาก RAM ที่มีอยู่ โดยอาศัยเนื้อที่ฮาร์ดดิสก์เป็นที่พักข้อมูลเมื่อ RAM เต็ม กล่าวคือ ข้อมูลใน RAM บางส่วนจะถูกสลับ (swap) ไปไว้ที่ฮาร์ดดิสก์ เพื่อให้มีที่ว่างสำหรับใช้งาน ต่อเมื่อข้อมูลส่วนนั้นต้องใช้งาน จึงสลับกลับเข้ามาในหน่วยความจำเหมือนเดิม ด้วยวิธีการเช่นนี้ ทำให้ลินุกซ์สามารถอ้างหน่วยความจำได้เท่ากับขนาดของ RAM บวกด้วยขนาด swap space (ลบด้วย overhead นิดหน่อย)

ลินุกซ์สามารถใช้ไฟล์ปกติเป็น swap ก็ได้ แต่โดยทั่วไปจะสร้าง swap partition โดยเฉพาะ เพื่อประสิทธิภาพที่ดีกว่า ดังนั้น นอกเหนือจากพาร์ทิชันสำหรับระบบไฟล์รากของลินุกซ์แล้ว swap partition เป็นพาร์ทิชันแรกที่คุณควรสร้างแยกต่างหาก โดยใช้พาร์ทิชันชนิด Linux Swap (รหัส `0x82`)

สำหรับขนาดของ swap partition ถ้าเป็นเคอร์เนลรุ่นเก่า จะมีปัญหาในการ swap เมื่อ RAM ถูกใช้จนหมด ถ้ากำหนดขนาด swap น้อยกว่าขนาดของ RAM โดยจะเกิด page thrashing (คืออาการ swap อย่างต่อเนื่องยาวนาน ทำให้ฮาร์ดดิสก์ระรัวอยู่ตลอดกระบวนการ และระบบจะค้าง) ดังนั้น ขนาดของ swap partition ควรมากกว่าขนาดของ RAM ที่มี และสูตรสำหรับค่าที่เหมาะสมก็คือ สองเท่าของขนาดของ RAM

1.4.2 การอัพเกรดระบบ

คุณควรออกแบบเผื่อการอัพเกรดระบบไว้ด้วย ซึ่งในบางครั้งหมายถึงการฟอร์แมตฮาร์ดดิสก์แล้วเริ่มติดตั้งใหม่ คุณจึงไม่ควรเก็บข้อมูลของผู้ใช้ในพาร์ทิชันเดียวกับระบบ กล่าวคือ `/home` เป็นไดเรกทอรีที่คุณควรพิจารณาสร้างเป็นพาร์ทิชันแยกต่างหาก

การแยก `/home` มีประโยชน์มากในกรณีที่คุณติดตั้งลินุกซ์หลายระบบในเครื่องเดียวกันและต้องการใช้ข้อมูลร่วมกัน

1.4.3 การบูตเมื่อระบบขัดข้อง

อาการขัดข้องของระบบมีได้หลายระดับ แต่ระดับที่พื้นฐานที่สุดคือ ระบบไฟล์เสีย ซึ่งอาจเกิดจากการปิดเครื่องโดยไม่ได้ shutdown ตามขั้นตอน (ซึ่งในปัจจุบัน journaling ช่วยได้มาก แต่ก็ยังมีโอกาสที่จะพบปัญหา) หรือด้วยเหตุผลใดก็ตามแต่ พาร์ทิชันหนึ่งที่คุณควรจะปกป้องจากความเสียหายก็คือ `/boot` ซึ่งจะเก็บไฟล์ kernel image และข้อมูลต่างๆ ของ boot loader เพื่อที่อย่างน้อย คุณก็ยังสามารถบูตระบบเพื่อแก้ปัญหาได้

`/boot` จึงเป็นอีกไดเรกทอรีหนึ่งที่ควรพิจารณาแยกออกมาเป็นพาร์ทิชันต่างหาก คุณอาจป้องกันข้อมูลโดยไม่ต้อง mount ขณะระบบทำงานเลย คือใช้เป็นพาร์ทิชันสำหรับการบูตเท่านั้นจริงๆ และจะ mount ก็ต่อเมื่อต้องการอัพเกรดเคอร์เนลหรือ boot loader คุณทำเช่นนั้นได้โดยใช้ตัวเลือก `noauto` สำหรับ `/boot` ใน `/etc/fstab`

ขนาดของ `/boot` โดยปกติจะมีขนาดไม่เกิน 10 MB

1.4.4 การสำรองข้อมูล

หากคุณกำลังติดตั้งลินุกซ์เพื่อใช้กับระบบที่ให้บริการที่สำคัญ คุณอาจพิจารณาเรื่องความสะดวกในการสำรองข้อมูลเพิ่มขึ้นอีก นอกจาก `/home` ซึ่งเป็นข้อมูลของผู้ใช้แล้ว `/var` เป็นอีกไดเรกทอรีหนึ่งที่คุณอาจสำรองบ่อย เพราะเป็นข้อมูลส่วนที่มีความเปลี่ยนแปลงอยู่ตลอดเวลา ข้อมูลที่อยู่ใน `/var` มักจะได้แก่ ถุงเมล์ของผู้ใช้, web page, FTP site, CVS repository, log file ต่างๆ ฯลฯ

อีกไดเรกทอรีหนึ่งที่แม้จะเปลี่ยนแปลงไม่บ่อยเท่า แต่คุณต้องการสำรองข้อมูลไว้แน่ๆ คือ `/etc` เพราะเป็นที่เก็บ configuration ของระบบของคุณเอง ถ้าระบบของคุณพัง และต้องเริ่มติดตั้งใหม่ การสำรอง configuration จะช่วยประหยัดเวลาได้มาก โดยเฉพาะข้อมูลบัญชีผู้ใช้ในระบบ

ในระบบที่ให้บริการที่สำคัญ มักจะแยกพาร์ทิชันของระบบไฟล์ต่างๆ ตามลำดับความสำคัญ เพื่อที่จะลดความเสี่ยงของการพังพร้อมกันทั้งระบบ และอาจจะเพื่อเลือกอุปกรณ์ที่ใช้เก็บตามความวิกฤตของข้อมูล (เช่น อาจใช้ RAID กับส่วนที่วิกฤต)

วิธีหนึ่งก็คือ แบ่งพาร์ทิชันให้ไดเรกทอรีรากเก็บระบบพื้นฐานที่เพียงพอสำหรับการซ่อมแซมระบบเท่านั้น และอาจจะ mount เป็นแบบ read-only เพื่อป้องกันการเปลี่ยนแปลงที่ไม่ตั้งใจ ไดเรกทอรีที่จะอยู่กับพาร์ทิชันนี้ได้แก่ `/bin`, `/lib`, `/sbin`, `/dev` เป็นไดเรกทอรีที่อาจไม่จำเป็นต้องสำรองข้อมูลก็ได้ ถ้ามีซีดีรอมสำหรับติดตั้งใหม่ จากนั้น จึงเลือกสร้างพาร์ทิชันเพื่อเก็บส่วนอื่นๆ ของระบบ เช่น

`/usr` และ `/opt` เก็บโปรแกรมที่ใช้ใน multi-user mode ซึ่งเป็นส่วนที่ไม่ค่อยเปลี่ยนแปลงเช่นกัน (อาจ mount read-only ก็ได้ และอาจไม่จำเป็นต้องสำรอง ถ้ามีซีดีรอมสำหรับติดตั้งใหม่) แต่มีความวิกฤตน้อยกว่าระบบพื้นฐาน

`/home` เก็บข้อมูลของผู้ใช้ ถือว่าวิกฤตและสำรองบ่อย

`/var` ที่เก็บข้อมูลของบริการต่างๆ ของระบบ สำรองบ่อยพอประมาณ

`/etc` เก็บ configuration ของระบบ สำรองเฉพาะเมื่อมีการ config ระบบ หรือเพิ่มผู้ใช้ใหม่

`/tmp` เป็นส่วนที่เปลี่ยนแปลงบ่อย ไม่วิกฤต ไม่ต้องสำรอง

1.4.5 ขนาดของโปรแกรมทั้งหมดที่ติดตั้ง

1.5 การติดตั้งบูตโหลดเดอร์

1.5.1 บูตโหลดเดอร์สำหรับลินุกซ์

มีหลายวิธีที่จะบูตเพื่อโหลดลินุกซ์ และก็มีบูตโหลดเดอร์หลายโปรแกรมที่ถูกพัฒนาขึ้นเพื่อใช้ในสถานการณ์ต่างๆ ตัวอย่างเช่น

- **SYSLINUX** เป็นบูตโหลดเดอร์อย่างง่าย ตัวมันเองติดตั้งใน boot sector ของแผ่นฟลอปปี้ได้ และโหลดลินุกซ์จากระบบไฟล์แบบ FAT ได้เท่านั้น เหมาะสำหรับทำแผ่นบูตลินุกซ์ด้วยฟลอปปี้แบบ MS-DOS

- **LOADLIN** ทำงานบนดอสหรือวินโดวส์ สามารถโหลดลินุกซ์จากระบบไฟล์ของลินุกซ์ได้ จึงสามารถใช้โหลดลินุกซ์ผ่านระบบบูตของดอสได้ โดยตั้งค่าใน `CONFIG.SYS` หรือ `AUTOEXEC.BAT`

- **LILO (LInux LOader)** ออกแบบมาเพื่อโหลดลินุกซ์โดยเฉพาะ แต่ก็สามารถโหลดระบบปฏิบัติการอื่นด้วยการ chain load ได้ สามารถติดตั้งใน MBR หรือ boot sector ของพาร์ทิชันได้ มี boot prompt สำหรับรับตัวเลือกของ Linux kernel และสามารถทำเมนูสำหรับเลือกระบบปฏิบัติการได้

- **GRUB (GRand Unified Bootloader)** เดิมออกแบบมาเพื่อโหลด GNU/Hurd แต่ก็ได้ขยายความสามารถในการโหลดระบบปฏิบัติการที่หลากหลาย สามารถอ่าน kernel image จากระบบไฟล์แบบต่างๆ ได้ด้วยตัวเอง (ในขณะที่ LILO ใช้วิธีระบุตำแหน่งเซกเตอร์ในดิสก์โดยตรง และต้องสร้าง map ใหม่ทุกครั้งที่เปลี่ยนเคอร์เนล) GRUB สามารถติดตั้งได้ใน MBR หรือ boot sector ของพาร์ทิชัน สามารถทำเมนูสำหรับเลือกระบบปฏิบัติการได้ และมีความสามารถพิเศษที่สามารถรับคำสั่งบูตเป็นบรรทัดคำสั่งได้ด้วย ทำให้มีความยืดหยุ่นสูงในการบูตในกรณีพิเศษต่างๆ

- **CHOS** ออกแบบมาเพื่อเป็นอีกทางเลือกหนึ่งนอกเหนือจาก LILO สามารถทำงานในลักษณะเดียวกับ LILO มีเมนูเลือกระบบปฏิบัติการ และมี GUI สำหรับตั้งค่าด้วย

บูตโหลดเดอร์ที่นิยมใช้กันแพร่หลายที่สุดคือ LILO และ GRUB ซึ่งเราจะกล่าวถึงต่อไป

1.5.2 รูปแบบการติดตั้งบูตโหลดเดอร์

เราสามารถติดตั้งบูตโหลดเดอร์ไว้ที่ขั้นตอนต่างๆ ของการบูตได้หลายแบบ ตามแต่การแบ่งพาร์ทิชันสำหรับระบบปฏิบัติการต่างๆ ในฮาร์ดดิสก์ ถ้าเราเข้าใจขั้นตอนการบูตแล้ว ก็จะสามารถจัดการการบูตระบบต่างๆ ในเครื่อง รวมทั้งใช้โหลดเดอร์ที่มากับระบบต่างๆ ร่วมกันได้

หัวข้อนี้จะกล่าวถึงรูปแบบการติดตั้งบูตโหลดเดอร์บางแบบ เพื่อเป็นตัวอย่าง (ตำแหน่งที่ทำเครื่องหมายดอกจัน (*) หมายถึงตำแหน่งที่สร้างเมนูบูต)

โหลดลินุกซ์ผ่านดอส

MBR		Partition Boot		OS
DOS MBR	→	MS DOS*	→	COMMAND.COM
			→	LOADLIN
				↓
				Linux

การบูตจะผ่านกระบวนการบูตปกติของดอส แต่ใน `CONFIG.SYS` หรือ `AUTOEXEC.BAT` จะไปเรียก LOADLIN ขึ้นมาแทน แล้ว LOADLIN จะไปโหลดเคอร์เนลลินุกซ์จากพาร์ทิชันลินุกซ์อีกทอดหนึ่ง คุณสามารถใช้ `BOOT.SYS` เลือก OS ที่จะบูตได้

จะเห็นว่า วิธีนี้ไม่มีการเขียน MBR หรือ boot sector เลย

บูตโหลดเดอร์ในพาร์ทิชัน

MBR		Partition Boot		OS
DOS MBR	→	LILO/GRUB*	→	Linux
		↓		
		OS อื่น		

บูตผ่าน MBR ของดอสตามปกติ ซึ่งจะเรียก active partition ขึ้นมาบูต (ซึ่งมีได้เพียงหนึ่งพาร์ทิชัน) ดังนั้น เราจึงตั้งให้พาร์ทิชันที่ติดตั้ง LILO/GRUB เป็น active partition จากนั้น เป็นหน้าที่ของ LILO/GRUB ที่จะสร้างเมนูหรือรับคำสั่งบูต OS ต่างๆ ในเครื่อง หากจำเป็นต้องบูต OS ใดๆ ในเครื่องที่ LILO/GRUB ไม่สามารถโหลดได้ ก็จำเป็นต้องเซ็ตพาร์ทิชันของ OS นั้นให้ active ด้วยคำสั่ง `fdisk` เมื่อต้องการ

ใช้ NT boot manager

MBR		Partition Boot		OS
BOOTACTV*	→	LILO/GRUB	→	Linux
	→	OS อื่น		

บูตผ่าน boot manager ของ Windows NT หรือ OS/2 หรือ boot manager ใดๆ ที่ไม่สามารถโหลดลินุกซ์ได้ boot manager จะเรียกพาร์ทิชันที่ผู้ใช้เลือกขึ้นมาบูต สำหรับพาร์ทิชันของลินุกซ์ คุณสามารถติดตั้ง LILO/GRUB แบบไร้เมนูไว้ที่ boot sector เพื่อโหลดลินุกซ์โดยเฉพาะ

ใช้ LILO/GRUB เป็น boot manager หลัก

MBR		Partition Boot	OS
LILO/GRUB*	——————→		Linux
	→	OS อื่น	

ติดตั้ง LILO/GRUB ไว้ที่ MBR ซึ่งทำให้ LILO/GRUB ทำงานตั้งแต่ต้น ผู้ใช้สามารถเลือก OS จากเมนูของ LILO/GRUB ซึ่งถ้าเป็นลินุกซ์หรือ OS อื่นที่ LILO/GRUB โหลดได้ ก็จะโหลดโดยตรง หรือถ้าเป็น OS ที่โหลดโดยตรงไม่ได้ (เช่น Windows) ก็ต้อง chain load โดยอาศัย boot sector ของพาร์ทิชันนั้นๆ

บูตลินุกซ์อย่างเดียว

MBR	Partition Boot	OS
LILO/GRUB	⟶	Linux

สำหรับเครื่องที่ติดตั้งลินุกซ์เพียงชุดเดียว คุณสามารถตัดเมนูออกแล้วให้ LILO/GRUB โหลดลินุกซ์ทันทีได้

1.5.3 การปรับแต่ง LILO

การปรับแต่ง LILO

การทำงานของ LILO แบ่งออกเป็นสองส่วน คือตัวบูตโหลดเดอร์เองและโปรแกรมติดตั้งบูตโหลดเดอร์ การปรับแต่งบูตโหลดเดอร์จะทำผ่านโปรแกรมติดตั้งบูตโหลดเดอร์ซึ่งเรียกใช้ได้จากเชลล์ของลินุกซ์ คือ `/sbin/lilo`

โปรแกรม `lilo` ดังกล่าวจะติดตั้งบูตโหลดเดอร์ตามที่กำหนดไว้ในไฟล์ `/etc/lilo.conf` ดังนั้น เมื่อคุณจะปรับแต่งบูตโหลดเดอร์ คุณจึงต้องทำสองขั้นตอน คือแก้ไข `/etc/lilo.conf` และสั่ง `/sbin/lilo` เพื่อติดตั้งบูตโหลดเดอร์ใหม่ ต่อไปนี้เป็นตัวอย่างของไฟล์ `/etc/lilo.conf`

```
prompt
timeout=50
default=Linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
lba32

image=/boot/vmlinuz-2.4.18-14
        label=Linux
        initrd=/boot/initrd-2.4.18-14.img
        read-only
        root=/dev/hda3

other=/dev/hda1
        label=Windows
```

จากตัวอย่างข้างต้น ความหมายของแต่ละส่วนก็คือ

`prompt` ให้บูตโหลดเดอร์หยุดรับคำสั่งบูตจากผู้ใช้

`timeout=50` หยุดรอเพียง 5.0 วินาที หากไม่มีคำตอบจะบูตค่าดีฟอลต์

`default=Linux` เลือกบูตหัวข้อที่มี label เป็น `Linux` ถ้าไม่มีการเลือก

`boot=/dev/hda` ให้ติดตั้งบูตโหลดเดอร์ที่ MBR ของฮาร์ดดิสก์ IDE ลูกแรก (ถ้ากำหนดเป็นพาร์ทิชัน เช่น `/dev/hda1` จะติดตั้งที่ boot sector ของพาร์ทิชันที่กำหนด)

`map=/boot/map` ให้ใช้ไฟล์ที่กำหนดเป็น map สำหรับโหลด kernel (จะกล่าวต่อไป)

`install=/boot/boot.b` ติดตั้ง boot sector image ที่กำหนดลงใน boot device ที่ระบุใน "`boot=...`"

`message=/boot/message` ไฟล์ที่เก็บข้อความที่จะแสดงก่อน boot prompt อาจเป็นภาพกราฟิกส์ได้ถ้ากำหนด boot sector image (ในหัวข้อ "`install=...`") ที่โหลดกราฟิกส์

`lba32` ใช้ Logical Block Address เพื่อใช้กับฮาร์ดดิสก์ใหญ่ๆ (กล่าวคือ เพื่อให้อ่าน cylinder ที่อยู่ถัดจาก cylinder ที่ 1024 ได้)

`image=/boot/vmlinuz-2.4.18-14...` เป็นส่วนกำหนดเมนูของ `Linux` โดยให้อ่าน kernel image ที่ `/boot/vmlinuz-2.4.18-14`, ใช้ label ในเมนูว่า `Linux`, ใช้ RAM Drive image ที่ `/boot/initrd-2.4.18-14.img`, ให้ mount ระบบไฟล์รากแบบ read-only ขณะบูต (แต่หลังจากบูตเสร็จแล้ว ลินุกซ์ก็จะ mount ระบบไฟล์รากใหม่อีกครั้งตามที่กำหนดใน `/etc/fstab`), และท้ายที่สุด ให้ใช้ `/etc/hda3` เป็นระบบไฟล์ราก

`other=/dev/hda1` เป็นส่วนกำหนดเมนูของ `Windows` โดยจะไม่โหลด image ของระบบปฏิบัติการเอง แต่จะส่งต่อให้ boot sector ของพาร์ทิชันที่ 1 (`/dev/hda1`) เป็นผู้โหลด

การติดตั้งบูตโหลดเดอร์

เนื่องจากตัวบูตโหลดเดอร์ของ LILO ไม่รู้จักโครงสร้างระบบไฟล์ของลินุกซ์หรือของระบบปฏิบัติการใดทั้งสิ้น แต่จะโหลดเคอร์เนลของลินุกซ์ด้วยการอ่านจากเซกเตอร์ต่างๆ ของฮาร์ดดิสก์โดยตรง โดยอาศัย map ที่โปรแกรมติดตั้งบูตโหลดเดอร์ `lilo` เตรียมไว้ให้ ดังนั้น ทุกครั้งที่มีการเปลี่ยนแปลงเคอร์เนลหรือไฟล์ message จะต้องสั่ง `/sbin/lilo` เพื่อสร้าง map ใหม่ทุกครั้ง

```
# lilo
```

การส่งพารามิเตอร์ของการบูต

ตามปกติ หากปรับแต่ง LILO แบบไม่ใช้เมนู เมื่อบูตระบบ LILO จะถามผู้ใช้ผ่าน prompt '`boot:`' ผู้ใช้สามารถระบุระบบที่ต้องการบูตโดยพิมพ์ label (ที่กำหนดโดยคำสั่ง '`label=...`' ในรายการใน `lilo.conf`) ซึ่งเป็นโอกาสที่ผู้ใช้สามารถเพิ่ม parameter ให้กับเคอร์เนลได้ โดยพิมพ์ต่อท้าย label ได้เลย เช่น

```
boot: linux single
```

เป็นการบูตระบบเป็นแบบ single user (กรุณาศึกษา boot parameter ของเคอร์เนลลินุกซ์ได้จากเอกสารของเคอร์เนล หรือ BootPrompt-HOWTO)

แต่ถ้าคุณใช้ LILO แบบมีเมนู การเลือกเมนูก็เหมือนกับการพิมพ์ชื่อ label นั่นเอง สำหรับเมนูในโหมดตัวอักษร จะมี prompt '`boot:`' อยู่ใต้เมนู ซึ่งจะรับ label ของรายการเมนูที่ถูกเลือกในปัจจุบันมาเติมโดยอัตโนมัติ คุณสามารถเริ่มพิมพ์พารามิเตอร์ของการบูตต่อโดยเริ่มกดแป้น space bar ได้ทันที

แต่ถ้าเป็นเมนูในโหมดกราฟิก จะไม่แสดง prompt '`boot:`' ดังกล่าว คุณต้องกดปุ่ม `Ctrl-x` (สำหรับ lilo ที่เก่ากว่ารุ่น 22) หรือ `Ctrl-i` (สำหรับ lilo ตั้งแต่รุ่น 22 เป็นต้นมา) เพื่อเรียก prompt ดังกล่าวขึ้นมา จากนั้น คุณสามารถพิมพ์คำสั่งบูตได้ตามปกติ

การตั้งรหัสผ่าน

ความสามารถในการส่งพารามิเตอร์สำหรับการบูตดังกล่าวอาจช่วยคุณได้ในกรณีที่ระบบขัดข้อง คุณอาจบูตเข้าโหมด single user เพื่อแก้ไขได้ แต่ในอีกแง่หนึ่ง มันก็เป็นการเปิดโอกาสให้ผู้ที่สามารถเข้าถึงเครื่องของคุณควบคุมเครื่องด้วยการบูตที่ไม่ปกติได้ คุณจึงอาจพิจารณาตั้งรหัสผ่านสำหรับการบูตเครื่องที่สำคัญ

LILO อนุญาตให้คุณตั้งรหัสผ่านกำกับระบบปฏิบัติการแต่ละรายการใน `lilo.conf` โดยเพิ่มคำสั่ง

```
password=PASSWORD
```

ตัวอย่างเช่น

```
image=/boot/vmlinuz-2.4.18-14
        label=Linux
        initrd=/boot/initrd-2.4.18-14.img
        read-only
        root=/dev/hda3
        password=bootpass
        restricted
```

สำหรับคำสั่ง `restricted` นั้น จะบอก LILO ไม่ให้ถามรหัสผ่านสำหรับการบูตปกติ และจะถามก็ต่อเมื่อมีการเพิ่มพารามิเตอร์เท่านั้น

ข้อควรระวังก็คือ รหัสผ่านนี้จะเก็บเป็นข้อความโดยตรง ดังนั้น เพื่อความปลอดภัยของระบบ คุณไม่ควรตั้ง permission ของ `lilo.conf` ให้ผู้ใช้ปกติอ่านได้ (GRUB สามารถเข้ารหัสรหัสผ่านก่อนเก็บได้)

1.5.4 การปรับแต่ง GRUB

GRUB เป็นบูตโหลดเดอร์ที่ออกแบบมาให้บูตระบบปฏิบัติการทั่วๆ ไปบน PC ได้ โดยพยายามอ่านระบบไฟล์ของระบบต่างๆ ด้วยตัวเองด้วย ทำให้ขั้นตอนต่างๆ ลดลงจากการใช้ LILO พอสมควร แต่ด้วยความครอบจักรวาลของ GRUB จึงจำเป็นต้องทราบข้อตกลงบางอย่าง และวิธีใช้งานที่แตกต่างออกไป

การเรียกชื่ออุปกรณ์

เนื่องจาก GRUB ถูกออกแบบมาเพื่อใช้บูตระบบปฏิบัติการทั่วไปบนเครื่อง PC ไม่จำกัดเฉพาะลินุกซ์ จึงมีข้อตกลงเรื่องการเรียกชื่ออุปกรณ์ที่ต่างจากลินุกซ์ ตัวอย่างเช่น รูปแบบต่อไปนี้:

```
(fd0)
```

หมายถึงแผ่นฟลอปปี้ในไดรฟ์แรก ชื่ออุปกรณ์ของ GRUB จะใช้เครื่องหมายวงเล็บครอบไว้ ส่วน 'fd' หมายถึงแผ่นฟลอปปี้ และเลข '0' หมายถึงหมายเลขไดรฟ์ โดยเริ่มนับจากศูนย์ การอ้างอุปกรณ์เช่นนี้หมายถึงแผ่นฟลอปปี้ทั้งแผ่น แต่ถ้าจะอ้างอิงพาร์ทิชันของฮาร์ดดิสก์ จะใช้รูปแบบดังตัวอย่างต่อไปนี้

```
(hd0,0)
```

หมายถึงฮาร์ดดิสก์ลูกแรก (hd0) พาร์ทิชันแรก (,0) ซึ่งการนับพาร์ทิชันก็จะสงวนหมายเลข 0 ถึง 3 ไว้สำหรับ primary partition และเริ่มนับ logical partition ที่หมายเลข 4 เป็นต้นไป ตัวอย่างเช่น

```
(hd0,4)
```

หมายถึง logical partition แรกของฮาร์ดดิสก์ลูกแรก

GRUB ไม่มีการแยกชื่อฮาร์ดดิสก์ IDE และ SCSI ออกจากกันเหมือนในชื่ออุปกรณ์ของลินุกซ์ แต่จะกำหนดหมายเลขให้กับฮาร์ดดิสก์ทั้งหมดในเครื่องตามลำดับ ซึ่งโดยปกติแล้ว หมายเลข IDE จะมาก่อน SCSI แต่ก็อาจเปลี่ยนแปลงได้ถ้าคุณเปลี่ยนลำดับการบูตใน BIOS โดยสลับอุปกรณ์ IDE กับ SCSI

และเนื่องจาก GRUB มีความสามารถในการอ่านระบบไฟล์ต่างๆ คุณจึงอ้างชื่อไฟล์ในพาร์ทิชันได้โดยตรง เช่น

```
(hd0,0)/vmlinuz
```

หมายถึงไฟล์ '/vmlinuz' ในพาร์ทิชันแรกของฮาร์ดดิสก์ลูกแรก

การติดตั้งบูตโหลดเดอร์

เช่นเดียวกับ LILO และบูตโหลดเดอร์ทั่วไป GRUB ก็มีการแบ่งแยกระหว่างโปรแกรมติดตั้งกับตัวบูตโหลดเดอร์ แต่การแบ่งแยกของ GRUB จะไม่ใช่การแยกกันเด็ดขาด คุณจะสามารถพบบรรทัดคำสั่งของ GRUB ได้ทั้งในบูตโหลดเดอร์และในโปรแกรมติดตั้ง แต่ในการติดตั้งลินุกซ์ปกติจะตั้งค่าไว้ให้บูตโหลดเดอร์สร้างเมนูขึ้นมาแทนบรรทัดคำสั่ง คุณจึงมักไม่เห็นบรรทัดคำสั่งขณะบูต นอกเสียจากการติดตั้งจะไม่สมบูรณ์

คุณสามารถเรียกโปรแกรมติดตั้งบูตโหลดเดอร์ของ GRUB ในลินุกซ์ได้โดยเรียก

```
# /sbin/grub
Probing devices to guess BIOS drives. This may take a long time.
```

จากนั้นจะปรากฏหน้าจอบรรทัดคำสั่งของ GRUB

```
   GNU GRUB  version 0.93  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first
   word, TAB lists possible command completions.  Anywhere else
   TAB lists the possible completions of a device/filename. ]

grub>
```

บรรทัดคำสั่งของ GRUB สามารถใช้ TAB เติมส่วนที่เหลือได้เหมือนกับ bash และถ้าคุณกด TAB ในบรรทัดคำสั่งที่ว่างเปล่า GRUB ก็จะแสดงคำสั่งทั้งหมดที่เป็นไปได้

```
grub> TAB
 Possible commands are: blocklist boot cat chainloader clear cmp
 color configfile debug device displayapm displaymem dump embed
 find fstest geometry halt help hide impsprobe initrd install
 ioprobe kernel lock makeactive map md5crypt module modulenounzip
 pager partnew parttype password pause quit read reboot root
 rootnoverify savedefault serial setkey setup terminal terminfo
 testload testvbe unhide uppermem vbeprobe
```

คำสั่งที่ใช้ออกจาก grub นี้คือ quit

```
grub> quit
```

การเรียกโปรแกรมติดตั้งบูตโหลดเดอร์ของ GRUB จะมีความจำเป็นน้อยกว่าของ LILO เพราะตัวบูตโหลดเดอร์ของ GRUB มีความสามารถในการอ่านไฟล์ configuration จากระบบไฟล์ของลินุกซ์โดยตรง สำหรับ GRUB แล้ว ถ้าโปรแกรมติดตั้งลินุกซ์ของคุณติดตั้งบูตโหลดเดอร์ของ GRUB ให้เรียบร้อยแล้ว คุณจะเรียกตัวติดตั้งบูตโหลดเดอร์อีกก็ต่อเมื่อมีการย้ายบูตโหลดเดอร์ไปติดตั้งที่อื่น หรือเมื่อบูตโหลดเดอร์เสียหายเท่านั้น ถ้าเป็นกรณีอื่น เช่น มีการอัพเกรดเคอร์เนล หรือมีการเพิ่มเมนูสำหรับบูตระบบปฏิบัติการที่ติดตั้งใหม่ หรือจะเปลี่ยนสี เปลี่ยนภาพกราฟิกของเมนู ก็เพียงแต่แก้ configuration ก็เพียงพอ ไม่จำเป็นต้องเรียกโปรแกรมติดตั้งบูตโหลดเดอร์เพื่อสร้าง map ใหม่เหมือน LILO

ถ้าคุณยังไม่ได้ติดตั้ง GRUB ในระบบของคุณมาก่อน (เช่น คุณอาจจะเลือก LILO ไว้ในขณะติดตั้ง แต่เปลี่ยนใจจะใช้ GRUB ภายหลัง) เมื่อติดตั้งแพกเกจ grub ในระบบแล้ว อาจจะลองตรวจสอบในไดเรกทอรี /boot/grub ว่ามีไฟล์ stage1, *_stage1_5 และ stage2 หรือไม่ (ที่สำคัญคือ stage1, stage2 และ *_stage1_5 สำหรับระบบไฟล์ที่คุณใช้ในพาร์ทิชันต่างๆ เช่น e2fs_stage1_5 ถ้าเป็นลินุกซ์ปกติ) ถ้ายังไม่มี คุณจำเป็นต้องติดตั้งไฟล์เหล่านี้ก่อน โดยอาจก็อปปี้ไปจากไดเรกทอรี /usr/lib/grub/i386-pc จากนั้นเรียก /sbin/grub เพื่อสั่งงาน GRUB

ก่อนอื่นคุณต้องกำหนดตำแหน่งของข้อมูลของบูตโหลดเดอร์ กล่าวคือ พาร์ทิชันที่คุณเก็บบรรดาไฟล์ /boot/grub/* นั่นเอง ข้อมูลเหล่านี้จะถูกใช้ขณะบูต โดยโค้ดสั้นๆ ใน stage1 ที่ติดตั้งใน boot sector หรือ MBR จะโหลดส่วนที่เหลือนี้ขึ้นมาทำงานต่อ หากคุณไม่มั่นใจว่าอยู่ในพาร์ทิชันไหน คุณสามารถใช้คำสั่ง find เพื่อค้นหาได้

```
grub> find /boot/grub/stage1
 (hd0,1)
```

สมมติว่า /boot ของคุณอยู่ที่พาร์ทิชันที่สองของฮาร์ดดิสก์ลูกแรก

```
grub> root (hd0,1)
 Filesystem type is ext2fs, partition type 0x83
```

จากนั้นสั่งติดตั้ง stage1 ของบูตโหลดเดอร์ในอุปกรณ์ที่ต้องการ สมมติว่าคุณต้องการติดตั้งบูตโหลดเดอร์ใน MBR

```
grub> setup (hd0)
 Checking if "/boot/grub/stage1" exists... yes
 Checking if "/boot/grub/stage2" exists... yes
 Checking if "/boot/grub/e2fs_stage1_5" exists... yes
 Running "embed /boot/grub/e2fs_stage1_5 (hd0)"...  16 sectors
are embedded.
succeeded
 Running "install /boot/grub/stage1 (hd0) (hd0)1+16 p
(hd0,1)/boot/grub/stage2 /boot/grub/menu.lst"... succeeded
Done.
```

เมื่อติดตั้งเสร็จแล้ว ออกจาก GRUB shell ได้ด้วยคำสั่ง `quit`

```
grub> quit
```

ทั้งหมดนี้อาจสั่งผ่านบรรทัดคำสั่งในเชลล์ของลินุกซ์ได้ด้วยคำสั่ง `grub-install` ก็ได้

```
# grub-install /dev/hda
```

อย่างไรก็ดี ขั้นตอนอัตโนมัตินี้อาจมีปัญหาในบางเครื่อง เพราะ GRUB อาจ map อุปกรณ์ของ BIOS มาเป็นอุปกรณ์ของระบบผิดพลาดในบางกรณี และทำให้บูตเครื่องไม่ได้ จึงแนะนำให้ใช้วิธีสั่งเองจะดีกว่า

การตั้งค่า GRUB

คุณมีอิสระที่จะตั้งค่า GRUB ก่อนหรือหลังติดตั้งบูตโหลดเดอร์ก็ได้ หรือคุณอาจไม่ตั้งค่าอะไรเลยก็ยังได้ โดยหากคุณไม่ได้ตั้งค่า หรือไฟล์ configuration เสียหาย คุณจะได้ GRUB shell ที่คุณสามารถสั่งบูตได้ หากคุณตกอยู่ในสถานการณ์เช่นนั้น และพอรู้คำสั่งของ GRUB มาบ้าง ก็ยังสามารถเข้าใช้ระบบได้ เช่น คุณทราบว่ามีลินุกซ์ติดตั้งอยู่ที่พาร์ทิชันที่สองของฮาร์ดดิสก์ลูกแรก คุณสามารถสั่งบูตได้ดังนี้

```
grub> kernel (hd0,1)/boot/vmlinuz
   [Linux-bzImage, setup=0x1400, size=0xe28ca]

grub> boot
```

อย่าลืมว่า บรรทัดคำสั่งของ GRUB shell สามารถใช้ TAB เติมชื่อส่วนที่เหลือระหว่างป้อนข้อมูลได้ ดังนั้น ถึงคุณจะจำชื่อไฟล์ของเคอร์เนลไม่ได้ ก็ไม่เป็นปัญหา

หรือสมมติว่าคุณต้องการบูต Windows ที่พาร์ทิชันแรก

```
grub> root (hd0,0)
 Filesystem type is fat, partition type 0xc

grub> makeactive
```

```
grub> chainloader +1

grub> boot
```

เป็นการส่งต่อให้ boot sector ของ Windows โหลดระบบต่อ

แต่คุณไม่ยอมทำเช่นนี้ทุกครั้งที่บูตเครื่องแน่ๆ คุณสามารถเก็บคำสั่งบูตเหล่านี้เพื่อทำเป็นเมนูสำหรับบูตได้ โดยเขียนไฟล์ `/boot/grub/menu.lst` ตัวอย่างเช่น

```
default       0
timeout       5
color         light-gray/blue white/black

title         Linux
root          (hd0,1)
kernel        /boot/vmlinuz root=/dev/hda5 ro

title         Windows
root          (hd0,0)
makeactive
chainloader   +1
```

ส่วนต่างๆ ในตัวอย่างข้างต้นมีความหมายดังนี้

`default 0` ตัวเลือกโดยปริยายถ้าไม่มีการเลือกคือตัวเลือกแรก

`timeout 5` รอรับคำตอบเพียง 5 วินาที ถ้าไม่มีคำตอบจะบูตตามค่าดีฟอลต์

`color ...` ตั้งสีเมนู โดยมีรูปแบบเป็น '*foreground/background*' มีข้อมูลสองชุด ชุดแรกเป็นสีปกติ ชุดหลังเป็นสีของแถบเมนู
คุณอาจเปลี่ยนไปใช้ภาพกราฟิกแทนก็ได้ โดยใช้คำสั่ง `splashimage` แทน เช่น

```
splashimage (hd0,1)/boot/grub/splash.xpm.gz
```

`title ...` เป็นส่วนกำหนดรายการเมนู โดยค่าใน `title` เป็นข้อความในตัวเลือกของเมนู คำสั่งถัดๆ มาก็เป็นคำสั่ง สำหรับบูตระบบนั้นๆ นั่นเอง

นั่นคือทั้งหมดที่คุณต้องทำ ไม่จำเป็นต้องสั่งสร้าง map อีกเหมือน LILO การเปลี่ยนแปลง configuration จะมีผลกับการบูตครั้งต่อไปทันที

อย่างไรก็ดี ในบางระบบ คุณอาจพบว่า ถึงคุณสร้าง `/boot/grub/menu.lst` แล้ว เมื่อบูตระบบใหม่ก็ยังไม่มีเมนู แต่กลายเป็น prompt ของ GRUB shell แทน ในกรณีเช่นนี้ ขอให้คุณลองสร้าง symbolic link `/boot/grub/grub.conf` ชี้ไปยัง `/boot/grub/menu.lst` เพื่อแก้ปัญหา

การเปลี่ยนคำสั่งขณะบูต

คุณสามารถเปลี่ยนคำสั่งบูตในเมนูของ GRUB ชั่วคราวขณะบูตได้ โดยจะมีผลสำหรับการบูตครั้งนั้นเท่านั้น ไม่มีผลต่อการบูตครั้งต่อๆ ไป

ในเมนูบูตของ GRUB คุณสามารถกด 'e' เพื่อแก้ไขคำสั่งบูตของเมนูที่กำลังเลือก GRUB จะแสดงคำสั่งบูตสำหรับรายการนั้นๆ คุณสามารถเลือกบรรทัดที่จะแก้แล้วกด 'e' อีกครั้งเพื่อแก้บรรทัดนั้นๆ แล้วเคาะ Enter เพื่อจบการแก้ไขในแต่ละบรรทัด เมื่อแก้ไขทุกอย่างเรียบร้อย คุณก็กด 'b' เพื่อเริ่มบูต

แทนคำสั่ง 'e' GRUB รุ่นใหม่ๆ จะรับคำสั่ง 'a' เพื่อเป็นทางลัดไปสู่การแก้ไขบรรทัด "kernel" ซึ่งเกี่ยวกับการส่ง boot parameter ให้กับเคอร์เนลโดยตรง (กรุณาศึกษา boot parameter ของเคอร์เนลลินุกซ์ได้จากเอกสารของเคอร์เนล หรือ BootPrompt-HOWTO)

การตั้งรหัสผ่าน

ความอิสระในการเปลี่ยนคำสั่งบูตในบางครั้งกลายเป็นช่องโหว่ของระบบรักษาความปลอดภัย ทำให้ผู้ที่เข้าถึงเครื่องได้สามารถควบคุมเครื่องได้มากเกินไป หากเรื่องนี้สำคัญสำหรับคุณ คุณอาจป้องกันได้ด้วยการตั้งรหัสผ่านในบูตโหลดเดอร์

คำสั่ง `password` ใน `menu.lst` จะถามรหัสผ่านสำหรับการบูตที่ไม่ปกติ คุณอาจใช้ตัวเลือก `--md5` กำกับเพื่อเก็บรหัสผ่านในรูปที่เข้ารหัสไว้

```
password --md5 PASSWORD
```

โดย `PASSWORD` เป็นรหัสผ่านที่เข้ารหัสแล้ว หากคุณไม่ใช้ตัวเลือก `--md5` GRUB จะถือว่า `PASSWORD` คือรหัสผ่านที่ผู้ใช้ต้องป้อนโดยตรง

คุณสามารถเข้ารหัสรหัสผ่านของคุณก่อน โดยใช้คำสั่ง `md5crypt` ใน GRUB shell เช่น

```
grub> md5crypt
Password: **********  (ป้อนรหัสผ่าน)
Encrypted: $1$uC6uv/$AEsPcz2pCldSDMbIZi/Yt1
```

จากนั้นจึงตัดเอาข้อความที่เข้ารหัสแล้วไปแปะแทน `PASSWORD` ในคำสั่ง `password --md5` ดังกล่าว

คุณสามารถกำหนดให้ปกป้องรายการเมนูใดๆ ด้วยรหัสผ่านได้โดยแทรกคำสั่ง `lock` ในคำสั่งบูตของรายการนั้นๆ โดยควรแทรกไว้หลังคำสั่ง `title` ทันที เพื่อไม่ให้คำสั่งอื่นถูกกระทำก่อน ตัวอย่างเช่น

```
password --md5 $1$uC6uv/$AEsPcz2pCldSDMbIZi/Yt1

title          Linux
lock
root           (hd0,1)
kernel         /boot/vmlinuz root=/dev/hda5 ro
```

คุณอาจใช้คำสั่ง `password` แทน `lock` เพื่อกำหนดรหัสผ่านใหม่ให้กับเมนูแต่ละเมนูก็ได้ถ้าคุณต้องการรหัสผ่านที่ต่างกันสำหรับแต่ละรายการ

บทที่

2

การตั้งค่าของระบบ

หลังจากติดตั้งบูตโหลดเดอร์ เคอร์เนล และระบบพื้นฐานแล้ว ก็จำเป็นต้องตั้งค่าเฉพาะต่างๆ ของระบบ อันจะทำให้เครื่องแต่ละเครื่องทำหน้าที่เฉพาะต่างกันไป หัวข้อนี้จะกล่าวถึงการตั้งค่าที่จำเป็น

2.1 เขตเวลา

เพื่อที่จะให้เครื่องทุกเครื่องที่เชื่อมกันผ่านเครือข่ายทั่วโลกสามารถสื่อสารกันในเรื่องเวลาได้ตรงกันถึงแม้จะอยู่คนละเขตเวลา เวลาของยูนิกซ์จึงอ้างเป็นเวลามาตรฐานกรีนิช (GMT หรือ UTC) เสมอ แต่เพื่อความสะดวกในการติดต่อกับผู้ใช้ จึงมีรูปแบบการอ้างเวลาท้องถิ่น (local time) ด้วย คุณจึงจำเป็นต้องกำหนด*เขตเวลา (time zone)* ให้กับเครื่องของคุณเสมอ

อีกส่วนหนึ่งที่เกี่ยวข้องกับเวลาของเครื่องก็คือ การเก็บเวลาในนาฬิกาของเครื่อง (hardware clock) คุณสามารถเลือกเก็บเวลาเป็น UTC หรือเวลาท้องถิ่นก็ได้ เพียงแต่ต้องบอกให้ลินุกซ์รู้ เมื่อบูตระบบจึงจะตั้งเวลาของระบบได้ถูก โดยปกติแล้ว ถ้าคุณใช้เครื่องร่วมกับดอสหรือวินโดวส์ซึ่งถือว่าเวลาของเครื่องเป็นเวลาท้องถิ่นเสมอ คุณก็ควรเลือกเก็บเวลาท้องถิ่น

ในส่วนของการตั้งเขตเวลานั้น เป็นส่วนของ GNU C library ซึ่งลินุกซ์ทุกยี่ห้อต่างใช้เหมือนกันหมด วิธีเซ็ตจึงไม่ต่างกัน คือไฟล์ `/etc/localtime` จะเป็น symbolic link ไปยังไฟล์ที่เหมาะสมใน `/usr/share/zoneinfo` ซึ่งจะเก็บรายละเอียดของเขตเวลาแต่ละเขตทั่วโลก (ลินุกซ์บางยี่ห้ออาจต่างกันไปบ้างในรายละเอียด เช่น Red Hat เก็บค่า `ZONE` ใน `/etc/sysconfig/clock` แต่ผลสุดท้ายก็จะตั้งค่า `/etc/localtime` เหมือนกัน)

ส่วนข้อมูลของเวลาในนาฬิกาของเครื่องนั้น เป็นเรื่องของกระบวนการบูตระบบที่จะอ่านค่าจากนาฬิกาของเครื่องมาตั้งเวลาระบบ (หรืออาจจะเก็บค่าคืนให้กับนาฬิกาของระบบเมื่อ shutdown ด้วยก็ได้) ซึ่งค่อนข้างจะต่างกันไปในลินุกซ์แต่ละยี่ห้อ ตัวอย่างเช่น

- **Red Hat** `/etc/sysconfig/clock` (`UTC={true/false}`)
- **Debian** `/etc/default/rcS` (`UTC={true/false}`)
- **Gentoo** `/etc/rc.conf` (`CLOCK={UTC/local}`)

เขตเวลาของประเทศไทยคือ "Asia/Bangkok" ซึ่งเท่ากับ GMT+7 ชื่อเขตเวลาคือ ICT (Indo-China Time) เป็นเขตเวลาที่ใช้ร่วมกับประเทศลาว กัมพูชา และเวียดนาม โดยอ้างอิงกับเส้นลองจิจูด 105 องศาตะวันออก (ผ่านจังหวัดอุบลราชธานี)

2.2 เครือข่าย

หากจะกล่าวว่า อินเทอร์เน็ตเกิดมาจากยูนิกซ์ ก็คงไม่ผิด เพราะโปรโตคอล TCP/IP ที่ใช้เป็นโปรโตคอลหลักในอินเทอร์เน็ตก็เริ่มพัฒนาในยูนิกซ์ (ที่ UC Berkeley) เครือข่ายอินเทอร์เน็ตยุคแรกๆ ก็มีแต่ยูนิกซ์เป็นหลัก

ที่กล่าวเช่นนี้ ก็เพื่อจะบอกว่า การตั้งค่าเน็ตเวิร์คเป็นสิ่งที่อยู่คู่กับยูนิกซ์ซึ่งเป็นต้นแบบของลินุกซ์ แม้ว่าคุณจะติดตั้งแบบ stand-alone คุณก็ยังต้องมี loopback network อยู่นั่นเอง เพื่อให้ระบบทำงานได้สมบูรณ์

2.2.1 ชื่อเครื่อง

เพื่ออ้างถึงเครื่องในเครือข่ายในรูปแบบที่มนุษย์จำได้ง่าย ยูนิกซ์จะใช้ *ชื่อเครื่อง (host name)* แทนหมายเลข IP โดยมีกลไกการ map จากชื่อไปเป็นหมายเลข IP ซึ่งใช้โดย Internet Protocol ไว้บริการ

สำหรับเครือข่ายท้องถิ่นเล็กๆ เป็นการง่ายที่จะตั้งชื่อเครื่องต่างๆ ไม่ให้ซ้ำกัน รวมทั้งไม่ยากเกินความสามารถของมนุษย์ที่จะจดจำชื่อเครื่องต่างๆ แต่เมื่อเชื่อมต่อเครือข่ายออกสู่อินเทอร์เน็ต ซึ่งมีเครื่องอื่นๆ ทั่วโลกเชื่อมต่ออยู่จำนวนมหาศาล ชื่อเครื่องจะต้องถูกขยายด้วย *ชื่อโดเมน (domain name)* เพื่อระบุตัวเครือข่ายที่เชื่อม ทำให้ชื่อไม่ซ้ำกัน และอ้างถึงได้ง่าย

ชื่อเครื่องและชื่อโดเมนในลินุกซ์และยูนิกซ์ทั่วไป โดยปกติจะระบุไว้ที่ไฟล์ `/etc/hostname` แต่ในลินุกซ์บางยี่ห้ออาจมีการย้ายข้อมูลไปไว้ที่อื่น เช่น Red Hat จะเก็บชื่อเครื่องไว้ที่ `/etc/sysconfig/network` ชื่อเครื่องนี้ จะ map เป็นหมายเลข IP โดยอาศัยข้อมูลในไฟล์ `/etc/hosts` ซึ่งมีรูปแบบเป็น

```
IP-address      hostname      alias
```

ตัวอย่างเช่น

```
127.0.0.1       localhost
127.0.0.1       anubis.cairo.org  anubis
```

สังเกตว่า ถึงแม้เครื่องคุณจะไม่ได้เชื่อมต่อกับเครือข่ายใดเลย คุณก็ยังต้องตั้ง loopback network interface ชื่อ `localhost` เสมอ โดยหมายเลข IP ของ loopback นี้จะเป็นหมายเลข `127.0.0.1` ทั้งนี้เพราะบริการหลายอย่างของยูนิกซ์จะออกแบบมาให้ทำงานผ่านระบบเครือข่ายเสมอ เช่น คิวเครื่องพิมพ์, ระบบ X Window ฯลฯ โดยสามารถปรับให้ใช้กับเครื่องเดียวได้โดยผ่าน loopback

2.2.2 IP Adress

หมายเลข IP เป็นคุณสมบัติที่แปรเปลี่ยนได้หลายรูปแบบ ตามอุปกรณ์เครือข่ายที่ใช้ เช่น อาจจะเป็น ethernet, modem หรือ Wi-Fi แล้วแต่กรณี แต่ละอุปกรณ์เหล่านั้นจะถูกระบบจัดการผ่านสิ่งที่เรียกว่า interface การกำหนดหมายเลข IP จะเกิดขึ้นเมื่อมีการ "up" interface เหล่านั้น

ถ้าเครื่องของคุณเป็นเครื่องที่ต่อกับเครือข่ายผ่าน ethernet LAN card ตลอดเวลา (ซึ่งถือเป็นกรณีปกติ และโปรแกรมติดตั้งมักจะถามค่าของอุปกรณ์นี้เสมอ) interface ที่คุณจะ up เมื่อบูตเครื่องก็

คือ `eth0` (หมายถึง ethernet LAN card ใบแรก) ซึ่งคุณสามารถกำหนดหมายเลข IP แบบตายตัว (static IP) หรือจะถามจากเครื่องที่ทำหน้าที่จัดสรรหมายเลข IP ของเครื่องต่างๆ ในเครือข่ายผ่าน DHCP (Dynamic Host Configuration Protocol) ก็ได้

การตั้งหมายเลข IP แบบตายตัว จะใช้คำสั่ง `ifconfig` ซึ่งมีรูปแบบการเรียกดังนี้

```
ifconfig iface ip-address [netmask mask] [broadcast broadcast]
```

ตัวอย่างเช่น

```
# ifconfig eth0 192.168.0.2
```

หากเป็นเครื่องลูกข่ายในเครือข่ายที่ใช้ DHCP ก็มีเครื่องมือสำหรับตั้ง IP ที่เด่นๆ อยู่สองโปรแกรม คือ `dhclient` และ `pump` โดยเพียงแต่สั่งคำสั่งใดคำสั่งหนึ่งดังกล่าวโดยไม่ต้องระบุอาร์กิวเมนต์ มันก็จะ broadcast ร้องขอไปในเครือข่าย เครื่องที่เป็นเซิร์ฟเวอร์จะตอบรับและแจกจ่ายหมายเลข IP มาให้

เพื่อที่จะให้กระบวนการดังกล่าวเป็นอัตโนมัติ ซึ่งค่อนข้างจำเป็นสำหรับการกำหนด IP แบบตายตัว ซึ่งมีค่าต่างๆ ค่อนข้างยุ่งยาก ลินุกซ์ยี่ห้อต่างๆ จึงสร้างกระบวนการที่ทำให้คุณสามารถสั่งเช่นนี้ได้

```
# ifup eth0   # เปิดใช้
# ifdown eth0   # ปิด
```

ซึ่งวิธีการตั้งค่าปกติของ interface ต่างๆ จะต่างกันไปในลินุกซ์แต่ละยี่ห้อ ตัวอย่างเช่น

1. **Red Hat** ตั้งค่าที่ `/etc/sysconfig/network-scripts/ifcfg-`*`iface`* สำหรับ interface *`iface`* ใดๆ
2. **Debian** ตั้งค่าที่ `/etc/network/interfaces` สำหรับทุก interface
3. **Gentoo** ตั้งค่าที่ `/etc/conf.d/net` สำหรับทุก interface

ในไฟล์ต่างๆ เหล่านี้ จะมีตัวเลือกให้ตั้งค่าว่าจะเปิดใช้ interface ใดขณะบูตด้วย

2.2.3 Gateway

ในระบบอินเทอร์เน็ต การเชื่อมต่อระหว่างเครื่องข้ามเครือข่ายจะอาศัยการกำหนดเส้นทาง (route) เพื่อส่งต่อแพ็กเก็ต (packet) เป็นทอดๆ จากเครื่องต้นทางสู่ปลายทาง สำหรับเครือข่ายที่ย่อยที่สุด ก็จะมีเครื่องหรืออุปกรณ์ที่ทำหน้าที่เป็น *เราเตอร์ (router)* หรือ *เกตเวย์ (gateway)* ในการส่งต่อแพกเก็ตระหว่างเครื่องลูกข่ายกับเครือข่ายระดับบนขึ้นไปที่เชื่อมต่ออยู่ (เช่น ISP) จากนั้น ISP จะมีระบบการหาเส้นทางที่ดีที่สุดไปยังเครื่องปลายทางอีกต่อหนึ่ง ซึ่งอยู่นอกเหนือขอบเขตของเอกสารนี้

ถ้าคุณกำลังติดตั้งเครื่องลูกข่ายที่มีการติดตั้งเกตเวย์เอาไว้แล้ว และคุณต้องการเชื่อมต่อกับอินเทอร์เน็ต คุณก็ต้องตั้งค่าหมายเลข IP ของเกตเวย์ให้แต่ละ interface ด้วย นอกเหนือจากการตั้งหมายเลข IP

คำสั่งสำหรับการตั้งเกตเวย์คือ `route` เช่น

```
# route add default gw 192.168.0.1
```

เป็นการกำหนดให้ใช้เครื่องหมายเลข `192.168.0.1` เป็นเกตเวย์ปกติสำหรับทุก interface

การตั้งค่าเกตเวย์อัตโนมัติ โดยปกติจะสามารถตั้งได้ในไฟล์เดียวกับที่ตั้งค่าหมายเลข IP ตัวอย่างเช่น

Red Hat: `/etc/sysconfig/network-scripts/ifcfg-eth0`:

```
DEVICE=eth0
IPADDR=192.168.0.2
GATEWAY=192.168.0.1
ONBOOT=yes
```

Debian: `/etc/network/interfaces`:

```
# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.

auto lo eth0

iface lo inet loopback

iface eth0 inet static
    address 192.168.0.2
    gateway 192.168.0.1
    netmask 255.255.255.0
```

Gentoo: `/etc/conf.d/net`:

```
iface_eth0="192.168.0.2 broadcast 192.168.0.255 netmask 255.255.255.0"
gateway="eth0/192.168.0.1"
```

2.2.4 DNS

ดังที่ได้กล่าวไปแล้ว ว่าการอ้างถึงเครื่องในอินเทอร์เน็ตสามารถใช้ชื่อเครื่อง (host name) แทนหมายเลข IP ได้ แต่การที่จะแปลงชื่อเครื่องเป็นหมายเลข IP ได้ ก็ต้องมีฐานข้อมูลของชื่อเครื่องต่างๆ ด้วย

เมื่อมีการอ้างชื่อเครื่อง ระบบสามารถค้นฐานข้อมูลดังกล่าวได้จากไฟล์ `/etc/hosts` ในเครื่องเอง และจากเครื่องที่ให้บริการแปลงชื่อ ซึ่งเรียกว่า *Domain Name Server (DNS)* ตามลำดับ (ลำดับนี้เป็นไปตามค่าที่กำหนดไว้ในไฟล์ `/etc/nsswitch.conf` ในหัวข้อ "`hosts:`" ซึ่งสามารถเปลี่ยนแปลงได้)

แน่นอนว่าคุณอาจเก็บชื่อเครื่องต่างๆ ในเครือข่ายของคุณเองไว้ในไฟล์ `/etc/hosts` ได้ และคุณอาจเก็บชื่อเครื่องบางเครื่องนอกเครือข่ายด้วย แต่คุณจะต้องการ DNS เพื่อฐานข้อมูลที่สมบูรณ์และมีการปรับปรุงตามสภาพจริงเมื่อต้องการแปลงชื่อเครื่องใดๆ ในอินเทอร์เน็ต

คุณสามารถลิสต์รายการ DNS ตามลำดับของการค้นหาได้ที่ไฟล์ `/etc/resolv.conf` เช่น

```
search thai.net
nameserver 202.44.202.2
nameserver 202.44.202.3
```

หัวข้อ `search` มีไว้ระบุ default domain ในการค้นหา ซึ่งทำให้คุณสามารถอ้างชื่อในโดเมนนั้นเฉยๆ โดยไม่ต้องระบุโดเมนได้ ส่วนหัวข้อ `nameserver` ก็เป็นรายการ DNS ที่จะค้นหาตามลำดับ ค่าในตัวอย่างเป็นค่าที่เหมาะกับเครือข่ายที่ผู้เขียนใช้อยู่เท่านั้น คุณควรสอบถาม ISP ของคุณถึงหมายเลข DNS ที่จัดเตรียมไว้ให้

คุณสามารถทดสอบ DNS ได้ด้วยการลองแปลงชื่อของเครื่องที่คุณรู้จักด้วยคำสั่ง `nslookup` หรือ `host` แล้วแต่ระบบของคุณ (ปัจจุบัน `nslookup` จะค่อยๆ ใช้น้อยลง และเปลี่ยนมาใช้ `host` แทน)

```
thep@anubis:~$ nslookup linux.thai.net
Note:  nslookup is deprecated and may be removed from future rele
ases.
Consider using the 'dig' or 'host' programs instead.  Run nslooku
p with
the '-sil[ent]' option to prevent this message from appearing.
Server:         202.44.202.2
Address:        202.44.202.2#53

Name:   linux.thai.net
Address: 164.115.5.2

thep@anubis:~$ host linux.thai.net
linux.thai.net          A       164.115.5.2
```

2.3 Mount Table

ระบบไฟล์ของยูนิกซ์สามารถ "mount" อุปกรณ์ต่างๆ เข้าในส่วนต่างๆ ของระบบไฟล์ที่มีเพียงรากเดียวได้ (ต่างจากดอสหรือวินโดวส์ที่หนึ่งอุปกรณ์คือหนึ่งราก) คุณสามารถสั่ง mount อุปกรณ์ได้ด้วยคำสั่ง `mount` ซึ่งมีรูปแบบ

```
mount device mount-point
```

ตัวอย่างเช่น

```
# mount /dev/hdb1 /home/ftp
```

เป็นการ mount พาร์ทิชันแรกของ primary slave IDE harddisk เข้ากับไดเรกทอรี `/home/ftp` ทำให้การเข้าถึง `/home/ftp` ในระบบหมายถึงรากของพาร์ทิชันดังกล่าว

คุณสามารถกำหนด default mount point สำหรับอุปกรณ์ต่างๆ ในไฟล์ `/etc/fstab` เพื่อที่คุณสามารถจะลดคำสั่งเหลือเพียงเท่านี้ได้

```
# mount /home/ftp
```

รูปแบบของไฟล์ `/etc/fstab` คือ

```
mount-point mount-point  type    options           dump pass
```

ตัวอย่างเช่น

```
/dev/hb1    /home/ftp    ext2    noauto,defaults   0    1
```

ไฟล์ /etc/fstab นี้ จำเป็นสำหรับการบูตระบบ เพราะเมื่อโหลดเคอร์เนลขึ้นมาทำงานแล้ว ระบบจะต้อง mount รากของระบบไฟล์ (/) เพื่อโหลดไฟล์ต่างๆ การ mount นี้จะอาศัยตารางใน /etc/fstab ซึ่งโหลดขึ้นมาในการ mount ชั่วคราวขณะโหลดเคอร์เนล โดยจะ mount ทุกรายการที่ไม่ได้ระบุ "noauto" ใน option ซึ่งมักจะได้แก่ไดเรกทอรีของระบบและ /home นั่นเอง

ดังนั้น เมื่อคุณแบ่งพาร์ทิชันแล้ว คุณจึงต้องกำหนด mount point ด้วย หากคุณติดตั้งด้วยโปรแกรมติดตั้งทั่วไป คุณจะสามารถกำหนด mount point ขณะแบ่งพาร์ทิชันได้ หรือสามารถเลือกกำหนด mount point สำหรับพาร์ทิชันที่แบ่งไว้แล้วก็ได้ โปรแกรมติดตั้งจะสร้างไฟล์ /etc/fstab ให้คุณเอง

2.4 บัญชีผู้ใช้

เมื่อติดตั้งระบบ บัญชีผู้ใช้ที่ต้องมีโดยอัตโนมัติก็คือ root หรือผู้ดูแลระบบนั่นเอง โปรแกรมติดตั้งส่วนใหญ่จะให้คุณตั้งรหัสผ่านของ root เสมอ และมันจะเป็นรหัสผ่านที่คุณจะใช้ล็อกอินเข้าระบบถ้าคุณไม่ได้เพิ่มบัญชีผู้ใช้อื่น

อย่างไรก็ดี คุณควรเพิ่มบัญชีผู้ใช้ปกติอีกหนึ่งบัญชีไว้ใช้งานปกติเสมอ การเข้าระบบด้วยบัญชี root ตลอดเวลาเป็นเรื่องที่เสี่ยงต่อการทำลายข้อมูลโดยไม่ตั้งใจ เปรียบเสมือนการใส่เสื้อเบอร์ S สวมกางเกงในทับกางเกงปกติอยู่ตลอดเวลา คุณสามารถทำลายข้าวของ หรือถูกกับดักในโปรแกรมที่มีไวรัสหลอกให้แพร่เชื้อ (เหมือนที่เกิดในระบบปฏิบัติการบางระบบ) ได้ตลอดเวลา

โปรแกรมติดตั้งปกติจะให้คุณสร้างบัญชีผู้ใช้ได้ขณะติดตั้ง คุณจึงควรใช้ประโยชน์ตรงนี้ แต่หากคุณต้องการเพิ่มบัญชีผู้ใช้ภายหลัง คุณสามารถทำได้โดยใช้คำสั่ง useradd เช่น

```
# useradd somsri
```

และตั้งรหัสผ่านเริ่มต้นให้กับผู้ใช้ด้วยคำสั่ง passwd

```
# passwd somsri
Enter new UNIX password:(ป้อนรหัสผ่านใหม่ ซึ่งจะไม่ปรากฏบนจอภาพ)
Retype new UNIX password:  (ป้อนรหัสผ่านใหม่ซ้ำอีกครั้งเพื่อยืนยัน)
passwd: password updated successfully
```

บทที่

3

การติดตั้งโปรแกรม

ด้วยโปรแกรมติดตั้งของลินุกซ์ในปัจจุบัน คุณสามารถเลือกโปรแกรมที่ต้องการติดตั้งในเครื่องได้ แต่เมื่อติดตั้งเสร็จแล้ว คุณมักจะต้องการติดตั้งโปรแกรมเพิ่มเติมอีก วิธีติดตั้งโปรแกรมจะแตกต่างกันไปตามระบบแพกเกจของลินุกซ์ที่คุณใช้ แต่สิ่งที่คุณมั่นใจได้ก็คือ ในแต่ละระบบที่คุณใช้ โปรแกรมทุกโปรแกรมที่คุณติดตั้งจะใช้วิธีการเดียวกันหมด และจะประพฤติตามระเบียบกฎเกณฑ์ที่กำหนดอย่างคงเส้นคงวา จะไม่แยกโปรแกรมติดตั้ง แยกเมนู แยกวิธีรื้อถอน อย่างที่คุณอาจพบในระบบปฏิบัติการอื่น อีกทั้งการมีระบบแพกเกจที่แน่นอน บวกกับการไม่มีข้อจำกัดเรื่อง license ทำให้โปรแกรมต่างๆ ที่ติดตั้งในลินุกซ์สามารถใช้ไลบรารีต่างๆ ร่วมกันได้อย่างมีประสิทธิภาพ ลดความซ้ำซ้อนของทรัพยากร

ระบบแพกเกจของลินุกซ์ยี่ห้อต่างๆ อาจแบ่งได้เป็นประเภทดังนี้

3.1 การคอมไพล์จากซอร์ส

โลกของยูนิกซ์เป็นโลกของ portability แม้แต่ตัว OS เองก็ยังเขียนด้วยภาษา C ซึ่งเป็นภาษาระดับสูง สามารถคอมไพล์และติดตั้งได้ใน platform ต่างๆ วิธีติดตั้งโปรแกรมแบบดั้งเดิมของยูนิกซ์จึงเป็นการคอมไพล์โปรแกรมจากซอร์ส และนั่นก็คือโลกของการแจกจ่ายโปรแกรมในรูปซอร์สโค้ดตั้งแต่ยังไม่มีคำว่า ‘‘ซอฟต์แวร์เสรี” (Free Software) หรือ “โอเพนซอร์ส” เกิดขึ้นในโลกนี้

ยิ่งในยุคของ “ซอฟต์แวร์เสรี” และ “โอเพนซอร์ส” เช่นนี้ วิธีติดตั้งด้วยการคอมไพล์จากซอร์สจึงเป็นวิธีที่อยู่คู่กับลินุกซ์ ที่คุณสามารถใช้ได้กับลินุกซ์ทุกยี่ห้อ แต่ลินุกซ์ก็มีพัฒนาการของมัน วิธีคอมไพล์โปรแกรมก็ได้รับการพัฒนามาตามลำดับ จึงอาจจำแนกวิธีติดตั้งได้เป็น

1. **การคอมไพล์ด้วยมือ** เป็นการทำทุกอย่างด้วยตัวเอง ตั้งแต่ตรวจสอบไลบรารีต่างๆ ที่จำเป็นสำหรับการคอมไพล์ว่ามีครบหรือไม่ และต้องควบคุมลำดับก่อนหลังของการติดตั้งโปรแกรมต่างๆ ด้วยตัวเอง

2. **ระบบ Portage** พัฒนามาจากระบบ portage ของ FreeBSD ซึ่งช่วยลดภาระในการตรวจสอบไลบรารีที่จำเป็น โดยระบบ portage จะคอมไพล์และติดตั้งไลบรารีที่ขาดให้โดยอัตโนมัติ ตัวอย่าง

ของระบบนี้ก็คือ ระบบ GAR ที่ใช้คอมไพล์และติดตั้ง GNOME 2 (ชื่อว่า GARNOME) และ KDE 3 (ชื่อว่า konstruct) และระบบ emerge ของ Gentoo Linux

3.2 ระบบ Binary Package

การคอมไพล์จากซอร์สเป็นการใช้ซอร์สโค้ดของซอฟต์แวร์โอเพนซอร์สอย่างเต็มประสิทธิภาพก็จริง แต่ในหลายกรณีก็ไม่สะดวกที่จะคอมไพล์โปรแกรม เช่น ลินุกซ์สำหรับผู้ใช้ใหม่ หรือสำหรับผู้ใช้ทั่วไปที่ไม่ได้ต้องการแก้ไขโปรแกรม หรือแม้แต่สำหรับนักพัฒนาที่จะเลือกศึกษาและแก้ไขเฉพาะโปรแกรมที่สนใจ

ระบบ binary package จึงเริ่มพัฒนาขึ้น โดยมีรูปแบบหลักๆ สามรูปแบบคือ

1. **TGZ** หรือ **TBZ2** เป็น binary package แบบแรกๆ ที่ทำขึ้นจากการคอมไพล์โปรแกรมแล้วเลือกเอาเฉพาะไฟล์ผลลัพธ์มา `tar` รวมกันเข้า แล้วบีบอัดด้วย `gzip` (เป็น `*.tgz`) หรือ `bzip2` (เป็น `*.tbz2`) การติดตั้งก็เพียงแตกไฟล์ต่างๆ ลงในไดเรกทอรีรากของระบบ แต่ภาระในการตรวจสอบไลบรารีที่ต้องการยังคงเป็นของผู้ใช้ ตัวอย่างลินุกซ์ที่ใช้แพกเกจแบบนี้คือ Slackware

2. **RPM (Red Hat Package Management)** พัฒนาโดย Red Hat หลักการรวมไฟล์ผลลัพธ์จากการคอมไพล์ก็ยังคงเหมือนกับ TGZ แต่ได้เพิ่ม metadata ให้กับแพกเกจด้วย เช่น requires (แพกเกจอื่นที่ต้องติดตั้งก่อน), conflicts (แพกเกจที่ติดตั้งร่วมกันไม่ได้) รวมทั้งมีสคริปต์สำหรับปรับแต่งตั้งค่าในขั้นตอนต่างๆ ของการติดตั้งและรื้อถอนอีกด้วย RPM เป็น binary package ที่แพร่หลายที่สุดในปัจจุบัน ใช้ใน Red Hat, Mandrake, SuSE, TurboLinux, Conectiva รวมทั้ง Linux TLE, Linux SIS, และ GrandLinux ของไทยเราด้วย

3. **DEB (Debian Package)** พัฒนาโดย Debian ซึ่งเป็นลินุกซ์ที่เป็นสาธารณะมากที่สุดเท่าที่มีอยู่ ระบบแพกเกจของ Debian ก็นับว่าพัฒนาไปได้ไกลมากเมื่อเทียบกับระบบอื่น Debian package พัฒนามาในแนวทางเดียวกับ RPM แต่ metadata ของ Debian จะละเอียดกว่า เช่น dependency หลายระดับ โดยแบ่งเป็น depends, recommends, suggests ทำให้กำหนดความสัมพันธ์ระหว่างแพกเกจได้ตรงตามความเป็นจริงมากขึ้น รวมทั้ง policy ที่แน่นหนาของ Debian ก็ทำให้ระบบแพกเกจของ Debian มีความแข็งแกร่ง ไม่ค่อยเกิดปัญหาระหว่างแพกเกจ แม้จะพัฒนาโดยประชาคมที่ใหญ่โตก็ตาม แพกเกจแบบ DEB ใช้ใน Debian และ distribution อื่นที่พัฒนาไปจาก Debian เช่น Libranet, Xandros

นอกจากข้อดีในเรื่องการติดตั้งที่สะดวกแล้ว การใช้ binary package ยังช่วยให้แบ่งแพกเกจเป็นส่วนต่างๆ ที่เล็กลงได้ ทำให้ผู้ใช้สามารถเลือกติดตั้งเฉพาะส่วนที่จำเป็น เช่น ไม่จำเป็นต้องติดตั้งส่วน header file ของไลบรารีต่างๆ ถ้าต้องการเพียงแค่ให้โปรแกรมใช้งานโดยไม่ได้ต้องการคอมไพล์โปรแกรม เป็นต้น ในขณะที่การคอมไพล์จากซอร์สมักจะต้องติดตั้งทั้งหมด

แต่การคอมไพล์จากซอร์สก็มีข้อดีคือ สามารถปรับแต่งรูปแบบของการคอมไพล์เพื่อเปิด/ปิดความสามารถต่างๆ ได้ตามต้องการ หรือให้ optimize เข้ากับเครื่องมากที่สุด สามารถศึกษาโครงสร้างของโปรแกรมหรือหาความสามารถต่างๆ ที่ซ่อนอยู่ ทั้งยังสามารถแก้ไขโปรแกรมได้สะดวกเมื่อมีปัญหาหรืออยากได้ความสามารถเพิ่มเติม กล่าวโดยสรุปก็คือ ได้ใช้ปรัชญาโอเพนซอร์สอย่างเต็มที่

3.3 Advanced Package Management

ในบรรดาระบบติดตั้งที่กล่าวมาข้างต้น จะมีระดับการตรวจสอบสภาวะของระบบในระดับที่ต่างกันไป ในการคอมไพล์จากซอร์สนั้น ก่อนคอมไพล์มักจะมีสคริปต์สำหรับตรวจสอบหาไลบรารีต่างๆ ที่จำเป็นเสมอ

หากคุณปรับปรุงระบบให้ผ่านการตรวจสอบ โปรแกรมที่ติดตั้งก็จะสามารถคอมไพล์ผ่านและทำงานได้

แต่เมื่อทำเป็น binary package สิ่งที่ผู้รับได้รับมาจะไม่มีสคริปต์ตรวจสอบดังกล่าว ดังนั้น หากเป็นแพกเกจแบบ TGZ หน้าที่ในการตรวจสอบสภาวะของระบบจะเป็นหน้าที่ของผู้ใช้ที่จะต้องมีความรู้ถึงความต้องการของแพกเกจเอง แพกเกจอีกสองชนิด คือ RPM และ DEB จึงถูกพัฒนาขึ้น โดยเพิ่ม metadata กำกับแพกเกจ และระบบติดตั้งจะไม่ยอมให้ผ่านถ้าสภาวะของระบบไม่สอดคล้องกับเงื่อนไขที่กำหนด

อย่างไรก็ตาม แม้จะมี metadata ช่วยตรวจสอบ ทำให้ไม่ต้องคอยจำ dependency ของแพกเกจ แต่ความไม่สะดวกที่ยังคงเหลือก็คือ การเกิดลูกโซ่ของ dependency เป็นทอดๆ ทำให้ผู้ใช้ยังต้องติดตามหาแพกเกจที่จำเป็นมาติดตั้งให้ครบด้วยตนเอง โดยเฉพาะหากเป็นการดาวน์โหลดผ่านโมเด็มจะเป็นกรณีที่เลวร้ายที่สุด

Debian เป็น distribution แรกที่เริ่มแก้ปัญหานี้ โดยพัฒนาระบบ APT (Advanced Packaging Tool) มาครอบทับระบบ dpkg (Debian Package Manager) ที่ใช้ติดตั้งแพกเกจแบบ DEB ธรรมดาอีกชั้นหนึ่ง โดยสร้างฐานข้อมูลของแพกเกจ DEB ทั้งหมดพร้อมข้อมูลความสัมพันธ์ระหว่างแพกเกจต่างๆ ทำให้สามารถตรวจสอบแพกเกจที่จำเป็นทั้งหมดในการติดตั้งแพกเกจใดๆ และจัดการดาวน์โหลดและติดตั้งตามลำดับได้ในคราวเดียวได้

ระบบ APT เป็นข้อได้เปรียบของ Debian เหนือ distribution อื่นอยู่นานพอควร จนกระทั่ง Conectiva บริษัท distribution ลินุกซ์ของบราซิลแห่งหนึ่งซึ่งใช้ระบบ RPM ได้พอร์ทระบบ APT ของ Debian มาใช้กับ RPM กลายเป็นจุดเริ่มต้นของผู้ใช้ลินุกซ์ตระกูล RPM ที่ได้ใช้ระบบ APT และลินุกซ์เจ้าอื่นๆ ก็เริ่มใช้ตาม รวมทั้ง Linux TLE ของไทยด้วย (เริ่มตั้งแต่รุ่น 4.1) ในขณะที่ Mandrake มีระบบ urpmi เป็นของตัวเอง

อีกระบบหนึ่งที่คล้ายกับ APT แต่ใช้กับการคอมไพล์จากซอร์สก็คือ ระบบ GAR ที่ใช้ใน garnome (GNOME 2) และ konstruct (KDE 3) และระบบ emerge ของ Gentoo Linux ระบบดังกล่าวจะจัดลำดับการคอมไพล์แพกเกจต่างๆ ตามลำดับโดยอัตโนมัติ

ในที่นี้จะขอกล่าวถึงเพียงระบบ APT ที่ใช้กันค่อนข้างแพร่หลาย ตั้งแต่ Debian มาจนถึงลินุกซ์ที่ใช้ RPM บางเจ้า เช่น Conectiva และ Linux TLE หรือแม้แต่ผู้ใช้ Red Hat เอง ก็สามารถใช้ระบบ APT จากเว็บไซต์ `http://freshrpms.net`

3.3.1 การกำหนดแหล่งแพกเกจ

ระบบ APT จะอาศัยฐานข้อมูลแพกเกจที่มีการสร้างไว้เฉพาะ ดังนั้น คุณจึงต้องเลือกแหล่งแพกเกจที่สนับสนุน APT แล้วระบุแหล่งเหล่านั้นในไฟล์ `/etc/apt/sources.list` ซึ่งมีรูปแบบดังนี้

สำหรับ Debian:

```
deb     url dist components...
deb-src url dist components...
```

สำหรับ RPM:

```
rpm     url dist components...
rpm-src url dist components...
```

`deb` หรือ `rpm` ใช้สำหรับระบุแหล่ง binary package ส่วน `deb-src` หรือ `rpm-src` ใช้ระบุแหล่งของ source สำหรับ build binary package ในกรณีที่ต้องการแก้ไขโปรแกรมหรือตอมไพล์ใหม่จากซอร์สโค้ด ตัวอย่างเช่น สำหรับผู้ใช้ Red Hat 9 ที่ต้องการใช้ freshrpms:

```
rpm     http://ayo.freshrpms.net redhat/9/i386 os updates freshrpms
rpm-src http://ayo.freshrpms.net redhat/9/i386 os updates freshrpms
```

ในกรณีที่คุณต้องการใช้ซีดีรอมเป็นแหล่ง (ต้องเป็นซีดีรอมที่สนับสนุน APT ด้วย) ก็ทำได้โดยสั่ง

```
# apt-cdrom add
```

โปรแกรมจะบอกให้คุณใส่แผ่นซีดีรอม จากนั้นก็จะอ่านข้อมูลแพกเกจในซีดีรอมโดยอัตโนมัติ

3.3.2 การกำหนด Proxy

ในกรณีที่คุณอยู่ในระบบเครือข่ายองค์กรที่บังคับให้ติดต่อสู่โลกภายนอกผ่าน proxy เท่านั้น คุณก็ยังสามารถใช้ระบบ APT ได้ โดยกำหนดตัวแประระบบชื่อ `http_proxy` หรือ `ftp_proxy` เช่น

```
# export http_proxy=http://my.proxy.ip:8080
```

3.3.3 การปรับปรุงรายการแพกเกจ

ก่อนที่คุณจะติดตั้งโปรแกรม คุณจำเป็นต้องอ่านข้อมูลแพกเกจมาจากแหล่งเสียก่อน เพื่อที่ APT ที่เครื่องของคุณจะสามารถตรวจสอบความสัมพันธ์ระหว่างแพกเกจต่างๆ ได้ คำสั่งที่ใช้ก็คือ

```
# apt-get update
```

โดยปกติมักจะต้องสั่งด้วย root permission

ในกรณีที่แหล่งแพกเกจของคุณเป็นซีดีรอม ก็ไม่จำเป็นต้องสั่ง `apt-get update` หลังจาก `apt-cdrom add` อีก

3.3.4 การติดตั้งโปรแกรม

สมมติว่าคุณต้องการติดตั้งโปรแกรม `thailatex` หลังจากที่คุณปรับปรุงรายการแพกเกจแล้ว คุณก็เพียงแต่สั่ง

```
# apt-get install thailatex
```

ระบบจะตรวจสอบว่ามีแพกเกจอะไรบ้างที่ `thailatex` ต้องการแต่ยังขาดอยู่ แล้วดาวน์โหลดมาติดตั้งให้โดยอัตโนมัติ

คุณอาจใช้ตัวเลือก `-u` เพื่อให้ระบบหยุดถามก่อนในกรณีที่มีแพกเกจอื่นต้องลงเพิ่มก็ได้

```
# apt-get -u install thailatex
```

3.3.5 การอัพเกรดระบบ

ระบบ APT สามารถช่วยอำนวยความสะดวกในการปรับปรุงระบบของคุณให้ทันสมัย รวมทั้งช่วยเป็นช่องทางอัตโนมัติสำหรับผู้ดูแลแพกเกจที่จะกระจายแพกเกจรุ่นใหม่ๆ ที่ได้แก้ไขข้อผิดพลาดหรือเพิ่มความสามารถแล้วไปสู่ผู้ใช้ คุณจึงอาจพิจารณาอัพเกรดระบบเป็นระยะๆ

หลังจากที่ปรับปรุงรายการแพกเกจด้วย `apt-get update` แล้ว คุณก็เพียงแต่สั่ง

```
# apt-get -u upgrade
```

ตัวเลือก `-u` จะบอกให้ `apt-get` หยุดถามก่อนเริ่มดาวน์โหลดและติดตั้ง เพื่อให้คุณมีโอกาสตรวจสอบรายชื่อแพกเกจที่มีการปรับปรุงก่อน

หากคุณไม่ต้องการอัพเกรดระบบทั้งหมด ก็ยังสามารถสั่งอัพเกรดเฉพาะโปรแกรมที่ต้องการได้ด้วยคำสั่ง `apt-get install` *`package`* เหมือนการติดตั้งใหม่ คำสั่งดังกล่าวจะตรวจสอบแพกเกจที่ระบุว่าติดตั้งหรือยัง ถ้ายังก็จะติดตั้งใหม่ ถ้าติดตั้งแล้ว ก็จะตรวจสอบว่ามีรุ่นใหม่ที่แหล่งหรือไม่ ถ้ามีก็จะอัพเกรดให้

3.3.6 การอัพเกรดข้ามรุ่น

คำสั่ง `apt-get upgrade` จะอัพเกรดเฉพาะแพกเกจที่เพียงเปลี่ยนรุ่นโดยไม่ได้ต้องการแพกเกจใหม่ที่ยังไม่เคยติดตั้ง และไม่ได้มีผลให้ต้องลบแพกเกจอื่นทิ้ง ถ้ามีแพกเกจที่เปลี่ยนรุ่นโดยกระทบแพกเกจอื่นเช่นนั้นแล้ว APT จะรายงานให้ทราบว่าแพกเกจนั้นจะถูก “kept back” ซึ่งหากต้องการอัพเกรด จะต้องใช้คำสั่ง `apt-get install` *`package`* เพื่ออัพเกรดแพกเกจเป็นรายตัวไป

คำสั่ง `apt-get upgrade` จะใช้กับการอัพเกรดระบบตามปกติ ซึ่งเป็นการอัพเกรดเล็กๆ น้อยๆ ตามวาระเท่านั้น หากเป็นการอัพเกรดข้ามรุ่นของตัวลินุกซ์ทั้งระบบ จะใช้คำสั่ง

```
# apt-get dist-upgrade
```

ซึ่งจะอัพเกรดทุกสิ่งทุกอย่างโดยไม่มีข้อยกเว้น

3.3.7 การถอดถอนโปรแกรม

คำสั่งสำหรับถอดถอนโปรแกรมก็คือ

```
# apt-get remove package
```

เช่นเดียวกับการติดตั้ง ระบบ APT จะตรวจสอบ dependency ก่อนถอดถอนระบบด้วย หากแพกเกจที่ถอดถอนนั้น จำเป็นสำหรับการทำงานของแพกเกจอื่น ก็เท่ากับทำให้แพกเกจเหล่านั้นทำงานล้มเหลวไปด้วย ดังนั้น APT จะพยายามลบแพกเกจเหล่านั้นด้วย โดยจะหยุดถามคุณก่อน

3.3.8 การแก้ปัญหาเมื่อฐานข้อมูลแพกเกจขัดข้อง

มีบ้างบางครั้งที่ฐานข้อมูลของ APT อาจมีปัญหาเกี่ยวกับ dependency ระหว่างแพกเกจที่ไม่เหมาะสม อาจจะเป็นเพราะการ upgrade ที่ไม่สมบูรณ์ในบางแพกเกจ หากคุณพบปัญหาเช่นนั้น คุณสามารถใช้คำสั่งต่อไปนี้เพื่อบอก APT ให้คำนวณหาวิธีแก้

```
# apt-get -f install
```

3.3.9 การเคลียร์เนื้อที่ดาวน์โหลด

ในการติดตั้งโปรแกรมผ่านคำสั่ง `apt-get install` หรือ `apt-get upgrade` หรือ `apt-get dist-upgrade` จากแหล่งแพกเกจในอินเทอร์เน็ต APT จะดาวน์โหลดแพกเกจมาไว้ที่แหล่งพักในฮาร์ดดิสก์ ซึ่งอยู่ใต้ราก `/var/cache/apt/archives` โดยแพกเกจที่อยู่ระหว่างดาวน์โหลด จะพักไว้ที่ไดเรกทอรีย่อยชื่อ `partial` เพื่อที่จะ resume ได้ในการดาวน์โหลดครั้งต่อไปหากดาวน์โหลดได้ไม่ครบ เมื่อดาวน์โหลดเสร็จแล้วจึงจะย้ายขึ้นมาไว้ที่รากดังกล่าว ก่อนที่จะสั่งติดตั้งแพกเกจเป็นขั้นสุดท้าย

แต่ APT ก็ไม่ได้ลบแพกเกจที่ดาวน์โหลดมานั้นทิ้ง เพื่อที่ว่า หากต้องการใช้อีก ก็จะได้ไม่ต้องดาวน์โหลดใหม่ แต่นั่นก็หมายความว่า เนื้อที่ฮาร์ดดิสก์จะถูกใช้ไปเรื่อยๆ กับการติดตั้งหรืออัพเกรดแต่ละครั้ง เมื่อคุณต้องการเนื้อที่ในฮาร์ดดิสก์ในส่วนดังกล่าวคืนเพื่อใช้งาน ก็สามารถสั่งลบแพกเกจต่างๆ ที่ดาวน์โหลดไว้นั้นทิ้งได้ด้วยคำสั่ง

```
# apt-get clean
```

3.3.10 การค้นหาแพกเกจในแหล่ง

การติดตั้งด้วยคำสั่ง `apt-get install` นั้น คุณต้องระบุชื่อแพกเกจให้ตรงตามชื่อในแหล่งแพกเกจ คุณสามารถใช้ข้อมูลในฐานข้อมูลแพกเกจให้เป็นประโยชน์ได้ หรืออาจจะใช้ค้นหาโปรแกรมที่ต้องการสำหรับงานหนึ่งงานใดก็ได้

ก่อนอื่น คุณต้องสั่งปรับปรุงฐานข้อมูลแคชสำหรับการค้นหาเสียก่อน (อาจจะปรับปรุงทุกครั้งที่คุณสั่ง `apt-get update` ก็ได้) ด้วยคำสั่ง

```
# apt-cache gencaches
```

ส่วนคำสั่งที่ใช้ค้นหาก็คือ

```
# apt-cache search keyword
```

นอกจากนี้ คุณอาจใช้คำสั่ง `apt-cache` ในการแสดงข้อมูลรายละเอียดของแพกเกจก็ได้

```
# apt-cache show package
```

บทที่

4

ระบบ X Window

ระบบ GUI บนยูนิกซ์มีพื้นฐานมาจากระบบ X Window เหมือนกันหมด X Window เป็นระบบเปิด โปรแกรมต่างๆ จะอิงอาศัยโปรโตคอลที่เหมือนกัน ทำให้สะดวกต่อการย้ายระบบ หรือแม้แต่ทำงานร่วมกันผ่านระบบเครือข่ายได้ทันที ส่วนที่จะต่างกันของ X Window ในแต่ละแพล็ตฟอร์มจะเป็นเรื่องของไดรเวอร์ในระดับต่ำเสียเป็นส่วนใหญ่ สำหรับลินุกซ์และระบบอื่นๆ ที่คล้ายยูนิกซ์ซึ่งทำงานบนเครื่อง PC จะใช้ระบบ X Window ที่มาจากโครงการ XFree86 ซึ่งการตั้งค่าต่างๆ จะมีวิธีเฉพาะตัว

4.1 Configuration

XFree86 จะเก็บข้อมูลต่างๆ สำหรับการเริ่มทำงานของ X server ไว้ที่ไฟล์ `/etc/X11/XF86Config-4` หรือ `/etc/X11/XF86Config` โดยจะตรวจหาไฟล์ทั้งสองตามลำดับและใช้ไฟล์ที่พบก่อน

ลักษณะเนื้อหาในไฟล์ดังกล่าวจะแบ่งออกเป็น section บรรยายทรัพยากรต่างๆ ได้แก่

`Files` เก็บที่อยู่ของฟอนต์และข้อมูลสีต่างๆ

`Modules` เก็บโมดูลต่างๆ ที่จะโหลด

`ServerLayout` โยงไปถึง section `Screen` ซึ่งบรรยาย graphics card, จอภาพ และโหมดที่ใช้ และ section `InputDevice` ซึ่งบรรยายอุปกรณ์ข้อมูลเข้า เช่น แป้นพิมพ์ เมาส์

`Screen` โยงไปถึง section `Device` และ `Monitor` พร้อมกับกำหนดโหมดการแสดงผล

`Device` กำหนด driver สำหรับ graphics card

`Monitor` กำหนดรายละเอียดของจอภาพที่ใช้

`InputDevice` กำหนดอุปกรณ์ข้อมูลเข้าทั้งหลาย section นี้จะมีหนึ่งชุดต่ออุปกรณ์หนึ่งตัว

คุณสามารถอ่านคู่มือสำหรับรูปแบบไฟล์อย่างละเอียดได้จาก manual page `XF86Config(5x)`

```
thep@anubis~$ man XF86Config
```

4.2 โปรแกรมสำหรับตั้งค่า

XFree86 ได้เตรียมเครื่องมือสำหรับตรวจสอบระบบเพื่อสร้างไฟล์ XF86Config ที่เหมาะสมไว้ให้จำนวนหนึ่ง อย่างไรก็ดี distribution ต่างๆ ก็ได้เตรียมเครื่องมือแยกต่างหากเช่นกัน คุณอาจพิจารณาเลือกใช้ได้ตามความเหมาะสม

ในที่นี้จะขอกล่าวถึงเครื่องมือของ XFree86 ตั้งแต่เวอร์ชัน 4.0 เป็นต้นมา ซึ่งมีอยู่ในทุก distribution

4.2.1 การ probe ด้วย X Server เอง

X server ใน XFree86 มีความสามารถในการตรวจหาอุปกรณ์ในระดับหนึ่ง คุณสามารถใช้ X server เองสร้าง XF86Config เองได้ โดยคุณต้องแก้ไขเพิ่มเติมในส่วนที่ X server ไม่ได้ตรวจหาให้ เช่น แป้นพิมพ์ เมาส์ ฟอนต์

การสร้าง XF86Config ด้วย X server ทำได้โดยกำหนดตัวเลือก -configure

```
# X -configure
```

X server จะตรวจสอบ graphics card แล้วเขียนไฟล์ XF86Config.new เป็นไฟล์เริ่มต้นให้ในไดเรกทอรีปัจจุบัน

วิธีนี้เหมาะสำหรับผู้ที่เข้าใจในรูปแบบของไฟล์ XF86Config เป็นอย่างดี

4.2.2 xf86config

xf86config เป็นเครื่องมือสำหรับสร้างไฟล์ XF86Config ด้วยการตอบคำถามหรือเลือกตัวเลือกในโหมดข้อความ แต่จะไม่มีการตรวจหาอุปกรณ์ให้

คุณสามารถสั่งที่เชลล์เช่นนี้

```
# xf86config
```

โปรแกรมจะแสดงหน้าจอแรกเพื่ออธิบายการทำงานของโปรแกรม

```
This program will create a basic XF86Config file, based on menu
selections you make.

The XF86Config file usually resides in /usr/X11R6/etc/X11 or
/etc/X11. A sample XF86Config file is supplied with XFree86; it
is configured for a standard VGA card and monitor with 640x480
resolution. This program will ask for a pathname when it is ready
to write the file.

You can either take the sample XF86Config as a base and edit it
```

```
for your configuration, or let this program produce a base
XF86Config file for your configuration and fine-tune it.

Before continuing with this program, make sure you know what
video card you have, and preferably also the chipset it uses and
the amount of video memory on your video card. SuperProbe may be
able to help with this.

Press enter to continue, or ctrl-c to abort.
```

จากนั้นจะตั้งคำถามไล่เรียงไปจนครบทุกหัวข้อ ตั้งแต่ เมาส์ แป้นพิมพ์ (พร้อมภาษา) จอภาพ graphics card และโหมดการแสดงผล เมื่อเสร็จแล้วจะเขียนไฟล์ `XF86Config` ที่มี comment กำกับอย่างละเอียดเพื่อความสะดวกในการแก้ไขของคุณ

วิธีนี้เหมาะสำหรับผู้ที่มีความเข้าใจในเรื่องฮาร์ดแวร์และทราบ spec ของอุปกรณ์ในเครื่องเป็นอย่างดี (เพราะจะไม่มีการตรวจหาให้), ต้องการ `XF86Config` ที่ใช้งานได้ทันที และต้องการเลือกตัวเลือกต่างๆ ที่ครบครัน

4.2.3 xf86cfg

เป็นโปรแกรมตั้งค่า X server ในแบบเดียวกับ `xf86config` แต่ใช้ GUI เรียกได้ว่า เป็นโปรแกรมที่มาแทนคำสั่ง `XF86Setup` ที่เคยมีใน XFree86 3.3.x อย่างไรก็ดี โปรแกรมนี้ยังไม่สมบูรณ์นัก เนื่องจากอ้างอิงการตรวจหาฮาร์ดแวร์ด้วย `X -configure` ที่อาจยังให้ข้อมูลด้านอื่นไม่สมบูรณ์ จึงไม่เหมาะที่จะใช้ในการตั้งค่าครั้งแรก แต่อาจจะใช้ปรับค่าในภายหลังได้

4.3 แป้นพิมพ์

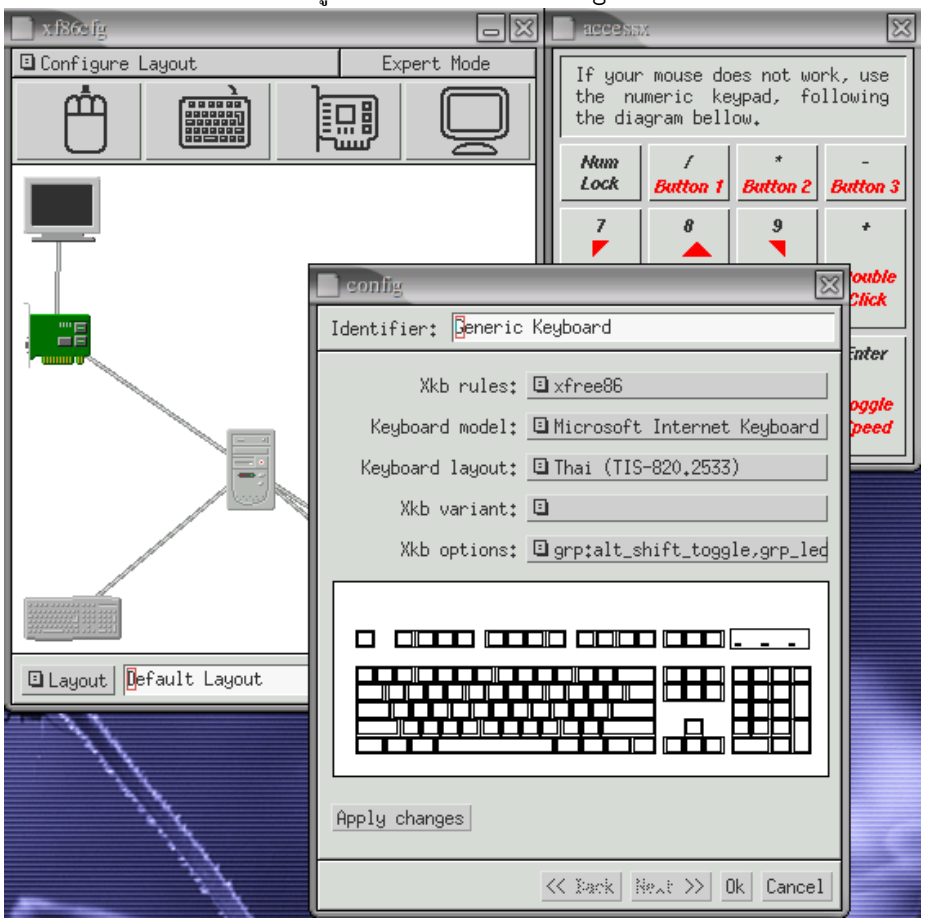
การตั้งค่าแป้นพิมพ์มีรายละเอียดที่เกี่ยวข้องหลายเรื่อง ที่สำคัญก็คือการกำหนดผังแป้นพิมพ์แบบต่างๆ การเลือกภาษา การตั้งการทำงานของปุ่มพิเศษและ LED ไปจนถึงกระบวนการรับข้อมูลของโปรแกรมเพื่อที่จะแปลงรหัสของปุ่มกดเป็นตัวอักษร

4.3.1 ผังแป้นพิมพ์

ส่วนประกอบใน `XF86Config` ที่เกี่ยวข้องกับแป้นพิมพ์ได้แก่ section `InputDevice` ซึ่งมีค่า `Driver` เป็น `"Keyboard"` ตัวอย่างเช่น

```
Section "InputDevice"
    Identifier "Generic Keyboard"
    Driver     "Keyboard"
    Option     "CoreKeyboard"
    Option     "XkbRules"   "xfree86"
    Option     "XkbModel"   "microsoftplus"
    Option     "XkbLayout"  "us,th"
```

รูปที่ 4.1: โปรแกรม xf86cfg



```
    Option      "XkbOptions" "grp:alt_shift_toggle,grp_led:scroll"
EndSection
```

ค่าต่างๆ ของแป้นพิมพ์จะอยู่ในบรรทัด `Option` ซึ่งแต่ละบรรทัดจะมีข้อมูลตามมาอีกสองคอลัมน์ เป็นชื่อตัวแปรและค่าของตัวแปรตามลำดับ โดยตัวแปรบางตัวซึ่งเป็น boolean สามารถละค่าได้ โดยจะหมายถึง `"true"`

ในตัวอย่างข้างต้น แต่ละบรรทัดมีความหมายดังนี้

`CoreKeyboard` เป็นค่า boolean หมายถึงการกำหนดให้รายการนี้เป็นแป้นพิมพ์หลัก ซึ่งจะมีแป้นพิมพ์หลักได้เพียงรายการเดียว ถ้ามีแป้นพิมพ์อื่นๆ และต้องการให้ใช้พร้อมกันได้ จะต้องตั้งค่า `SendCoreEvents` ในรายการนั้นด้วย (ทั้งนี้ ไม่รวมถึงการต่อแป้นพิมพ์ภายนอกของโน้ตบุค เพราะถือว่าต่อที่พอร์ทเดียวกับแป้นพิมพ์ภายในอยู่แล้ว)

`XkbRules`, `XkbModel` และ `XkbLayout` เป็นการเลือกภาษาของแป้นพิมพ์ โดยอาศัยกฎที่ชื่อ `xfree86` ซึ่งเป็นกฎที่ใช้เป็นค่าปกติสำหรับ XFree86 บนเครื่อง PC โดยป้อนค่าถัดมาสามค่าให้กับกฎดังกล่าวเพื่อประกอบเป็นผังแป้นพิมพ์ใหม่ ได้แก่

- `XkbModel` รุ่นของแป้นพิมพ์ โดยปกติสำหรับแป้นพิมพ์ในท้องตลาดปัจจุบันที่มีปุ่ม Windows และ Menu จะใช้ค่า `pc104` ในตัวอย่างข้างต้นใช้ `microsoftplus` เพื่อจะได้ใช้ปุ่ม multimedia และ internet shortcut ในแป้นพิมพ์ได้ (อย่างไรก็ดี ปุ่มพิเศษดังกล่าวยังไม่มีมาตรฐานตายตัว คุณอาจต้องลองหลายๆ ค่า ถ้าคุณมีแป้นพิมพ์ประเภทนี้และต้องการใช้ปุ่มพิเศษ)
- `XkbLayout` เลือกภาษาของแป้นพิมพ์ ในที่นี้ผู้เขียนใช้ XFree86 4.3.0 ซึ่งเริ่มแยกภาษาแป้นพิมพ์ออกเป็นผังละหนึ่งภาษา และเมื่อต้องการใช้ภาษาอังกฤษ (`us`) พร้อมกับไทย (`th`) จึงต้องผสมโดยใช้จุลภาคคั่น
  อนึ่ง หากคุณใช้ XFree86 ที่เก่ากว่า 4.3.0 คุณสามารถใช้เพียงผัง `th` ผังเดียวก็จะได้สองภาษา พร้อมกับปุ่มสลับภาษาโดยใช้ `Alt-LeftShift` เลือกภาษาอังกฤษ และ `Alt-RightShift` เลือกภาษาไทย
- `XkbOptions` ตัวเลือกพิเศษของผังแป้นพิมพ์ ในที่นี้ผู้เขียนได้เพิ่มตัวเลือกพิเศษสองตัวเลือก คือให้ใช้ปุ่ม `Alt-Shift` สลับภาษา (`grp:alt_shift_toggle`) และให้ใช้ LED ของ ScrollLock แสดงสถานะของภาษาปัจจุบัน (`grp_led:scroll`) โดยจะติดถ้ากำลังอยู่ในสถานะของภาษาที่สอง (ในที่นี้คือภาษาไทย)

4.3.2 ระบบอินพุตข้อความภาษาไทย

การเลือกผังแป้นพิมพ์ภาษาไทยเป็นเพียงส่วนหนึ่งของระบบการป้อนข้อมูลภาษาไทย โปรแกรมบน X Window โดยปกติจะรองรับระบบป้อนข้อความของ X ที่เรียกว่า XIM (X Input Method) ซึ่งสามารถปรับเปลี่ยนได้ตามท้องถิ่นต่างๆ สำหรับภาษาไทย คุณต้องกำหนดท้องถิ่น (locale – อ่านว่า “โลแคล”) ให้เป็นไทย เพื่อให้ระบบป้อนข้อความดังกล่าวทำงาน

ในบรรดาหัวข้อโลแคลต่างๆ (6 หัวข้อสำหรับมาตรฐาน POSIX) หัวข้อที่มีผลต่อ XIM คือ `LC_CTYPE`

วิธีตั้งโลแคลให้เป็นไทย (`th_TH`) ทำได้โดยตั้งตัวแปรระบบ `LANG` หรือ `LC_ALL` ซึ่งจะมีผลต่างกันดังนี้

1. หาก `LC_ALL` ถูกตั้งค่าไว้ จะมีผลบังคับใช้กับทุกหัวข้อทันที

2. มิฉะนั้น หาก LC_CTYPE ถูกตั้งค่าไว้ จะมีผลกับ XIM
3. มิฉะนั้น หาก LANG ถูกตั้งค่าไว้ จะมีผลกับทุกหัวข้อที่ยังไม่ตั้งค่า
4. มิฉะนั้น จะใช้ค่าโลแคลปกติของภาษา C คือ C

นอกจากโลแคลแล้ว XIM ไทยยังสามารถตั้งค่าอย่างละเอียดได้อีกถึง 3 ระดับ ซึ่งจะมีผลต่อความเข้มงวดในการตรวจสอบลำดับการพิมพ์ คุณสามารถตั้งค่าได้ตามต้องการโดยตั้งค่าตัวแปรระบบ XMODIFIERS ด้วยรูปแบบ

```
XMODIFIERS="@im=mode"
```

โดยที่ *mode* มีค่าที่เป็นไปได้คือ

1. Passthrough ไม่มีการตรวจลำดับการป้อนข้อมูล
2. BasicCheck ตรวจลำดับการป้อนข้อมูลแบบปกติ
3. Strict ตรวจลำดับการป้อนข้อมูลแบบเคร่งครัด

หากไม่มีการตั้งค่า ค่าปกติจะเป็น BasicCheck

อย่างไรก็ดี สำหรับโปรแกรมที่ใช้ GTK+ 2 ซึ่งได้แก่โปรแกรมใน GNOME 2 เป็นตัวอย่างนั้น GTK+ 2 มีวิธีจัดการระบบป้อนข้อมูลโดยอัตโนมัติตามภาษา ทำให้คุณไม่จำเป็นต้องตั้งค่าโลแคลเอง นอกจากนี้ GTK+ 2 ยังมีระบบป้อนข้อมูลเข้าแบบอื่นนอกเหนือจาก XIM ซึ่งคุณเลือกได้จาก context menu ของ text entry widget ของ GTK+ ซึ่งในขณะที่เขียนเอกสารฉบับนี้ ระบบป้อนข้อมูล (input method) สำหรับภาษาไทยที่ถูกต้องกำลังอยู่ในขั้นทดสอบก่อนรวมเข้าเป็นส่วนหนึ่งของ GTK+ คุณจะพบ input method แบบ “Thai (Broken)” ซึ่งไม่มีการตรวจสอบลำดับใดๆ ถ้าคุณใช้ GTK+ มาตรฐาน แต่คุณสามารถเลือก “X Input Method” ได้

4.4 เมาส์

ส่วนประกอบใน XF86Config ที่เกี่ยวข้องกับเมาส์ได้แก่ section InputDevice เฉพาะรายการที่ Driver มีค่าเป็นเป็น "Mouse" เช่น

```
Section "InputDevice"
    Identifier  "Touch Pad"
    Driver      "Mouse"
    Option      "CorePointer"
    Option      "Device"            "/dev/psaux"
    Option      "Protocol"          "PS/2"
    Option      "Emulate3Buttons"   "true"
EndSection
```

ค่าต่างๆ ของเมาส์อยู่ที่รายการ Option เช่นเคย ในตัวอย่างข้างต้น แต่ละบรรทัด Option มีความหมายดังนี้

CorePointer เป็นค่า boolean หมายถึงการกำหนดให้รายการนี้เป็นเมาส์หลัก ซึ่งจะมีเมาส์หลักได้เพียงรายการเดียว ถ้ามีเมาส์อื่นๆ และต้องการให้ใช้พร้อมกันได้ จะต้องตั้งค่า SendCoreEvents ในรายการนั้นด้วย (ทั้งนี้ ไม่รวมถึงการต่อเมาส์ที่พอร์ท PS/2 เพิ่มจาก touch pad หรือ think point ของโน้ตบุค เพราะถือว่าต่อที่พอร์ทเดียวกับอุปกรณ์ชี้ภายในอยู่แล้ว)

`Device` เป็นอุปกรณ์เมาส์เชื่อมต่ออยู่ เช่น ใน PC ทั่วไปที่ใช้ PS/2 รวมทั้งอุปกรณ์ชี้ของโน้ตบุค ก็จะต่อที่ `/dev/psaux` ถ้าเป็นเมาส์ที่ต่อที่ COM1 ก็จะเป็น `/dev/ttyS0` หรือเมาส์ USB ก็จะเป็น `/dev/input/mice`

`Protocol` หมายถึงโปรโตคอลที่เมาส์ใช้ส่งข้อมูล ซึ่งจะขึ้นอยู่กับแบบของเมาส์ เช่น ถ้าเป็นเมาส์สองปุ่มที่ต่อกับ PS/2 ก็ใช้โปรโตคอล `PS/2` ถ้าเป็น wheel mouse ก็จะใช้ `IMPS/2` เป็นต้น

`Emulate3Buttons` เป็นการจำลองปุ่มกลางของเมาส์ในกรณีที่ใช้เมาส์สองปุ่มโดยการกดทั้งสองปุ่มพร้อมกัน สำหรับ wheel mouse ทั่วไปไม่จำเป็นต้องจำลองปุ่มกลาง เพราะสามารถคลิกล้อเป็นปุ่มกลางได้อยู่แล้ว

สำหรับ wheel mouse ซึ่งมีข้อมูลการหมุนของล้อเพิ่มเข้ามา ในทางเทคนิคจะเรียกว่าเป็นการเคลื่อนที่ตามแกน Z และสามารถเพิ่ม `Option ZAxisMapping` ได้ดังนี้

```
Section "InputDevice"
    Driver      "Mouse"
    ...
    Option      "ZAxisMapping"  "4 5"
    ...
EndSection
```

เป็นการกำหนดให้เหตุการณ์การหมุนล้อไปทางลบกลายเป็นปุ่มที่ 4 และไปทางบวกเป็นปุ่มที่ 5 (ปุ่ม 1, 2 และ 3 ได้แก่ปุ่มซ้าย ปุ่มกลาง และปุ่มขวาตามลำดับ)